# QUESTION 7.



**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`) and 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 8.



**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 9.

**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:

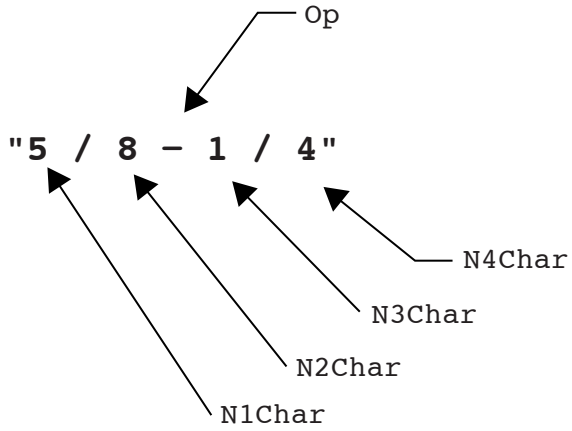
```
y ← ONECHAR("San Francisco", 6)
```
y ..... [1]

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.<br>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)** A program is to be written which accepts a string and then calculates a numeric value from this string. The input string and the calculated value are then to be sent to a remote computer over a communications link.

Study the following pseudocode:

```
OUTPUT "Enter string"
INPUT MyString
StringTotal ← 0

FOR i ← 1 TO CHARACTERCOUNT(MyString)
   NextNum ← ASC(ONECHAR(MyString, i))
   StringTotal ← StringTotal + NextNum
ENDFOR

OUTPUT MyString, StringTotal
```

Write the above pseudocode algorithm as **program code**.

*There is no need to show the declaration of variables or comment statements.*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[6]

**(c)** Explain the purpose of sending the value of `StringTotal` to the remote computer, in addition to `MyString`.

.....

.....

.....

.....[2]

# QUESTION 10.



**7** ASCII character codes are used to represent a single character.

Part of the code table is shown below.

**ASCII code table (part)**

| Character | Decimal | Character | Decimal | Character | Decimal |
|---|---|---|---|---|---|
| <Space> | 32 | I | 73 | R | 82 |
| A | 65 | J | 74 | S | 83 |
| B | 66 | K | 75 | T | 84 |
| C | 67 | L | 76 | U | 85 |
| D | 68 | M | 77 | V | 86 |
| E | 69 | N | 78 | W | 87 |
| F | 70 | O | 79 | X | 88 |
| G | 71 | P | 80 | Y | 89 |
| H | 72 | Q | 81 | Z | 90 |

Some pseudocode statements follow which use these built-in functions:

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

```
CHR(ThisInteger : INTEGER) RETURNS CHAR
```
returns the character with ASCII value `ThisInteger`.
For example: `CHR(66)` returns 'B'.

```
ASC(ThisCharacter : CHAR) RETURNS INTEGER
```
returns the ASCII value for character `ThisCharacter`.
For example: `ASC('B')` returns 66.

**(a)** Give the values assigned to the variables `A`, `B`, `C` and `D`.

The & operator is used to concatenate two strings.

The expression could generate an error; if so, write ERROR.

```
Num1 ← 5
A ← ASC('F') + Num1 + ASC('Z')
B ← CHR(89) & CHR(69) & CHR(83)
C ← CHARACTERCOUNT(B & "PLEASE")
D ← ASC(ONECHAR("CURRY SAUCE", 7))
```

**(i)** `A` ..... [1]

**(ii)** `B` ..... [1]

**(iii)** `C` ..... [1]

**(iv)** `D` ..... [1]

**(ii)** An experienced programmer suggests this pseudocode would be best designed as a 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 11.

**3** A string encryption function is needed. The encryption uses a simple character-sub

In this method, a new character substitutes for each character in the original string. This the encrypted string.

The substitution uses the 7-bit ASCII value for each character. This value is used as an index to 1D array, `Lookup`, which contains the substitute characters.
`Lookup` contains an entry for each of the ASCII characters. It may be assumed that the original string and the substitute characters are all printable.

For example:

- 'A' has ASCII value 65
- Array element with index 65 contains the character 'Y' (the substitute character)
- Therefore, 'Y' substitutes for 'A'
- There is a different substitute character for every ASCII value

The programmer writes a function, `EncryptString`, to return the encrypted string. This function will receive two parameters, the original, `PlainText` string and the 1D array.

**(a)** The first attempt at writing the pseudocode for this function is shown below.

Complete the pseudocode.

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

```
FUNCTION EncryptString(.............................) RETURNS STRING

    DECLARE ..................... , ...................... : CHAR

    DECLARE OldCharValue : ....................................

    DECLARE n : INTEGER

    DECLARE OutString : STRING


    ............................. //initialise the return string


    //loop through PlainText to produce OutString

    FOR n ← 1 TO ................. //from first to last character

       OldChar ← ...................//get next character

       OldCharValue ← ..............//find the ASCII value

       NewChar ← ...................//look up substitute character

       ..............................//concatenate to OutString

    ENDFOR


    ..........................................................
ENDFUNCTION
```

[10]

**(b)** Additional code needs to be written to allow the user to change some of the
array `Lookup`.

The user will input:

- the array start position
- the number of elements to change
- each new substitute character

At the end, the program will finally output a confirmation message.

The first version of the algorithm is represented by the flowchart on the following page.

**(i)** Write **program code** to declare the array `Lookup`.

Programming language .....

.....

.....

.....[2]

**(ii)** Write **program code** to implement the flowchart design.

In addition to the `Lookup` array, assume that the following variables have been declared:

```
StartPos, NumToChange, n, NewChar
```

Programming language .....

.....

.....

.....

.....

.....

.....

.....

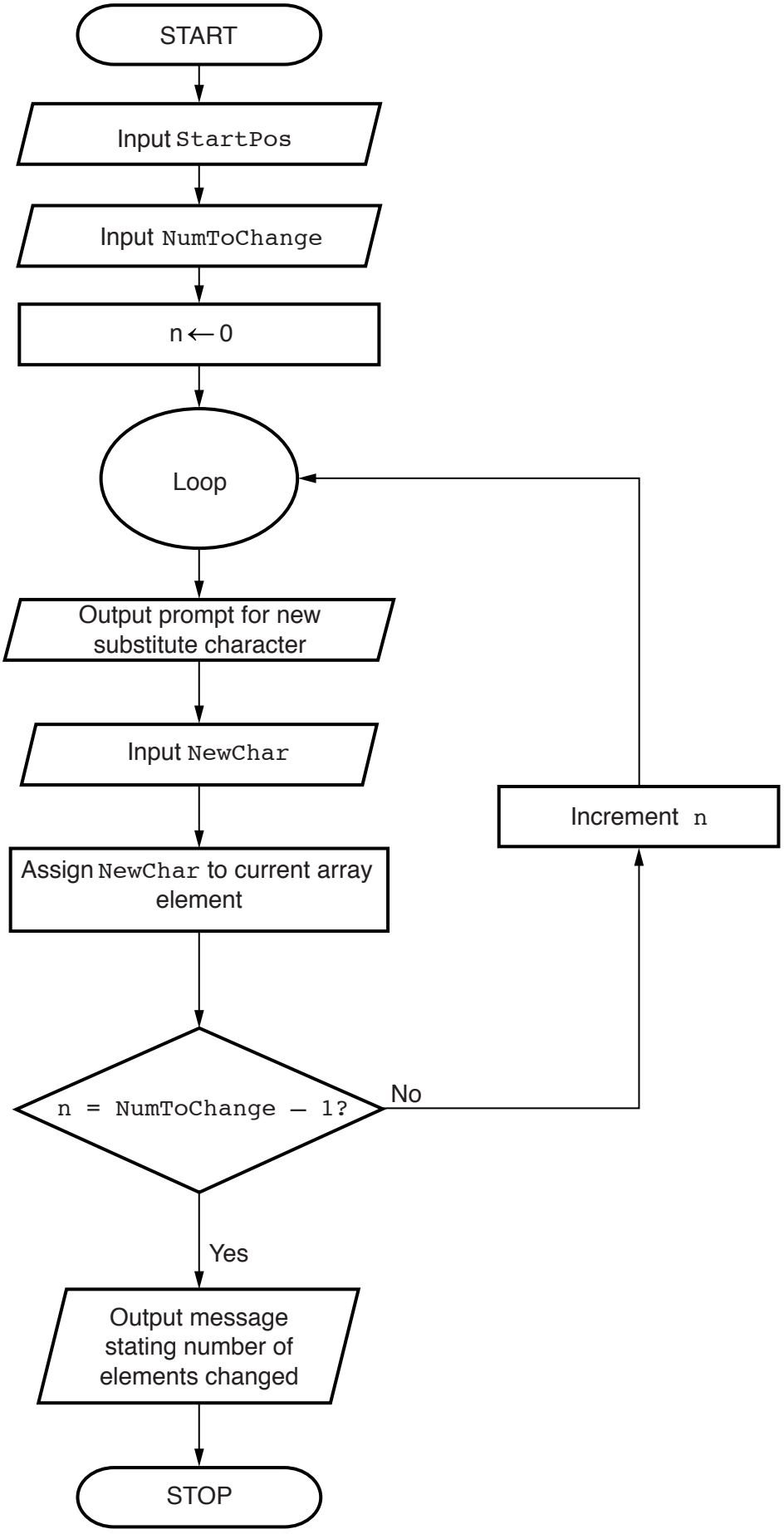
.....

.....

.....

.....

.....[6]


START
Input StartPos
Input NumToChange
n ← 0
Loop
Output prompt for new substitute character
Input NewChar
Assign NewChar to current array element
n = NumToChange – 1?
No
Increment n
Yes
Output message stating number of elements changed
STOP

# QUESTION 2.

**3** String encryption was implemented using a simple character-substitution met
`Decrypt`, is needed to reverse the encryption process and return the original chara

The encryption uses the 7-bit ASCII value for each character. This value is used as an in
1D array, `Lookup`, which contains the substitute characters.
`Lookup` contains an entry for each of the ASCII characters.

This function, `Decrypt`, will accept two parameters, a single character, `CipherChar`, and the 1D array, `Lookup`.

The steps involved in `Decrypt` are follows:

- Search for the character in the array
- Note the index value where the character is found (the index value is the ASCII value of the original character)
- Use the index value to obtain the original character

**(a)** The first attempt at writing the pseudocode for this function is shown below.

Complete the pseudocode.

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

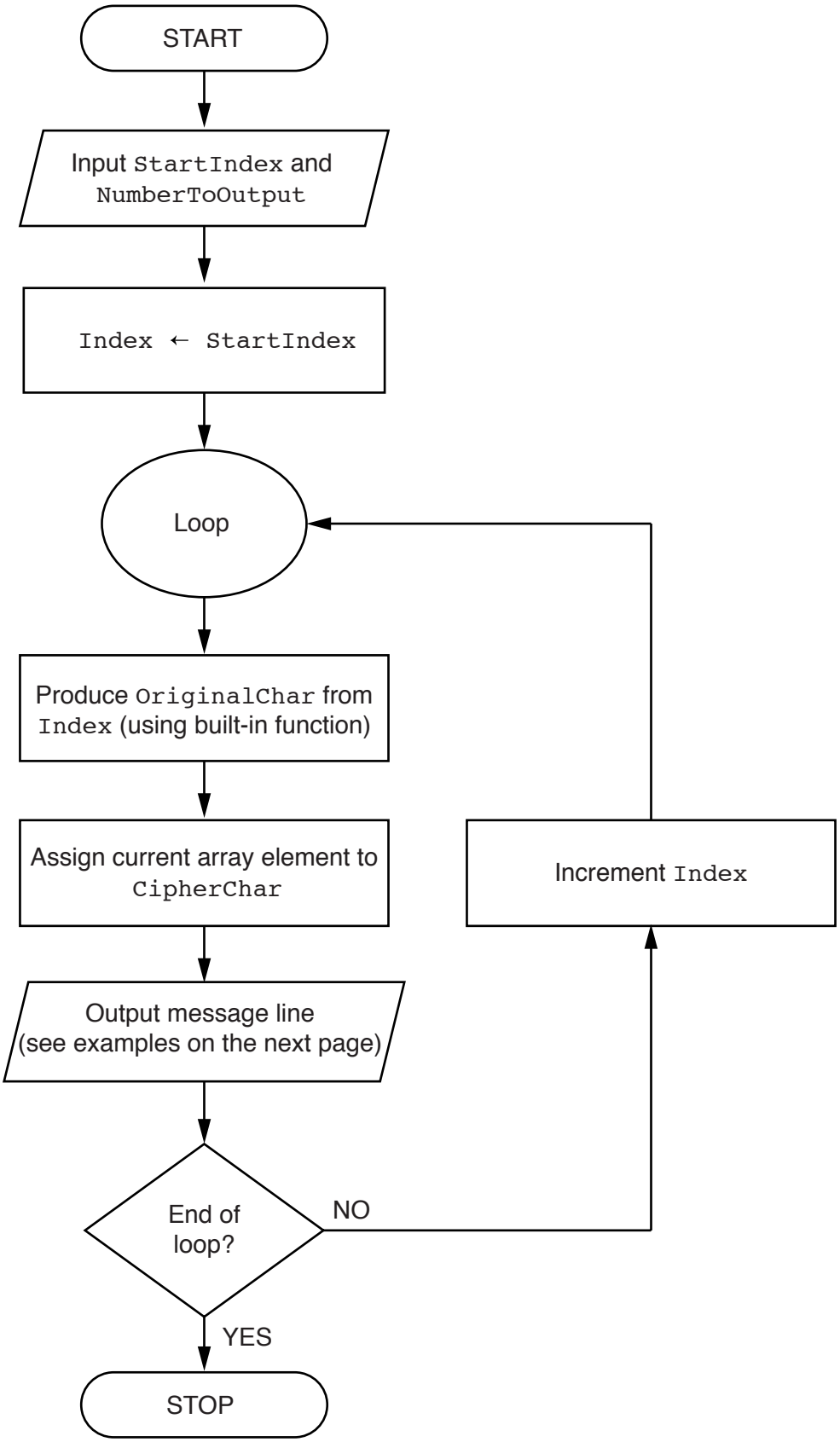
```
FUNCTION Decrypt(..... , .....) RETURNS CHAR

   DECLARE Found      : .....

   DECLARE ..... : INTEGER

   DECLARE OriginalChar      : CHAR

   Index ← 1         // .....

   Found ← FALSE
   //search for CipherChar in Lookup:

   WHILE .....
      //compare CipherChar with this array element:

      IF .....
         THEN
            .....//Set the flag
         ELSE
            Index .....//Move to next array element
      ENDIF
   ENDWHILE
   //dropped out of loop so must have found CipherChar:

   .....//convert Index to original character

   RETURN .....

ENDFUNCTION
```

[11]

**Question 3 continues on page 8.**

**(b)** A program is to be written to output part of the `Lookup` array.

The design of the algorithm is shown below.
It may be assumed that the characters output from `Lookup` are all printable.



For example, for the input of 65 and 3, the output will be:

```
Index 65: Character A has substitute character Y
Index 66: Character B has substitute character Q
Index 67: Character C has substitute character F
```

Write **program code** to implement the flowchart design.

In addition to the `Lookup` array, assume that the following variables have been declared:

`StartIndex, NumberToOutput, Index`

Programming language .....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

[6]

# QUESTION 3.



**4** A company employs Ahmed as a programmer.

**(a)** At College, before joining the company, Ahmed used two items of software for pro

- a text editor
- a compiler

Describe how he could have developed programs using these software tools.

Include in the description the terms 'object code' and 'source code'.

.....

.....

.....

.....

.....

.....

.....[3]

**(b)** Ahmed now uses an Integrated Development Environment (IDE) for programming.

**(i)** State **one** feature an IDE provides to help with the identification of syntax errors.

.....

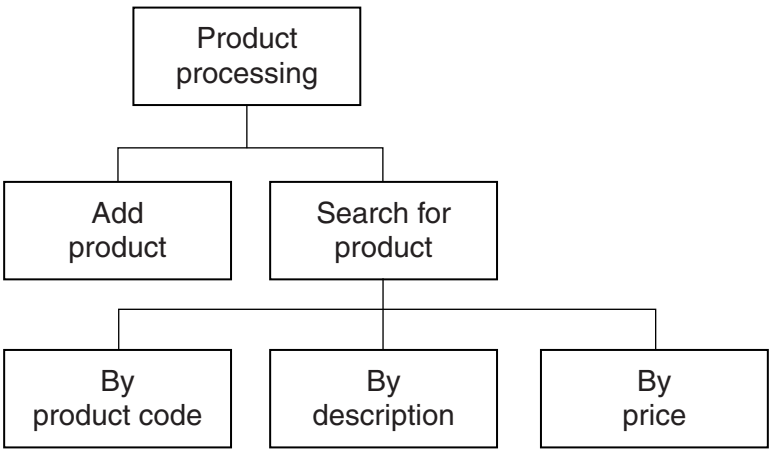
.....[1]

**(ii)** State **one** feature an IDE provides to carry out white box testing.

.....

.....[1]

**(c)** The company maintains a file of product data. Ahmed is to write a program to add a new product and search for a product based on the structure diagram shown:



The program records the following data for each product:

- product code
- product description
- product retail price

The text file `PRODUCTS` stores each data item on a separate line, as shown below:

File **PRODUCTS**

| |
|---|
| 0198 |
| Plums(10kg) |
| 11.50 |
| 0202 |
| Onions(20kg) |
| 10.00 |
| ~ |
| 0376 |
| Mango chutney(1kg) |
| 02.99 |
| ~ |
| 0014 |
| Mango(10kg) |
| 12.75 |

The program uses the variables shown in the identifier table.

| Identifier | Data type | Description |
|---|---|---|
| PRODUCTS | TEXT FILE | Storing the code, description and retail price for all current products |
| PCode | ARRAY[1:1000] OF STRING | Array storing the product codes |
| PDescription | ARRAY[1:1000] OF STRING | Array storing the product descriptions |
| PRetailPrice | ARRAY[1:1000] OF REAL | Array storing the product retail prices |
| i | INTEGER | Array index used by all three arrays |

**(i)** The first operation of the program is to read all the product data held i
and write them into the three 1D arrays.

Complete the pseudocode below.

```
OPEN .....

i ← 1

WHILE .....

  READFILE ("PRODUCTS", .....)

  READFILE ("PRODUCTS", .....)

  READFILE ("PRODUCTS", .....)

  .....

  .....

ENDWHILE

CLOSE "PRODUCTS"

OUTPUT "Product file contents written to arrays"
```

[5]

When Ahmed designed the `PRODUCTS` file, he considered the alternative file structure shown opposite.

It stores one product per line in the text file.

File **PRODUCTS**

| | |
|---|---|
| 0198 Plums(10kg) | 11.50 |
| 0202 Onions(20kg) | 10.00 |
| ~ | |
| 0376 Mango chutney(1kg) | 02.99 |
| ~ | |
| 0014 Mango(10kg) | 12.75 |

**(ii)** State **one** benefit and **one** drawback of this file design.

Benefit .....

.....

Drawback .....

.....[2]

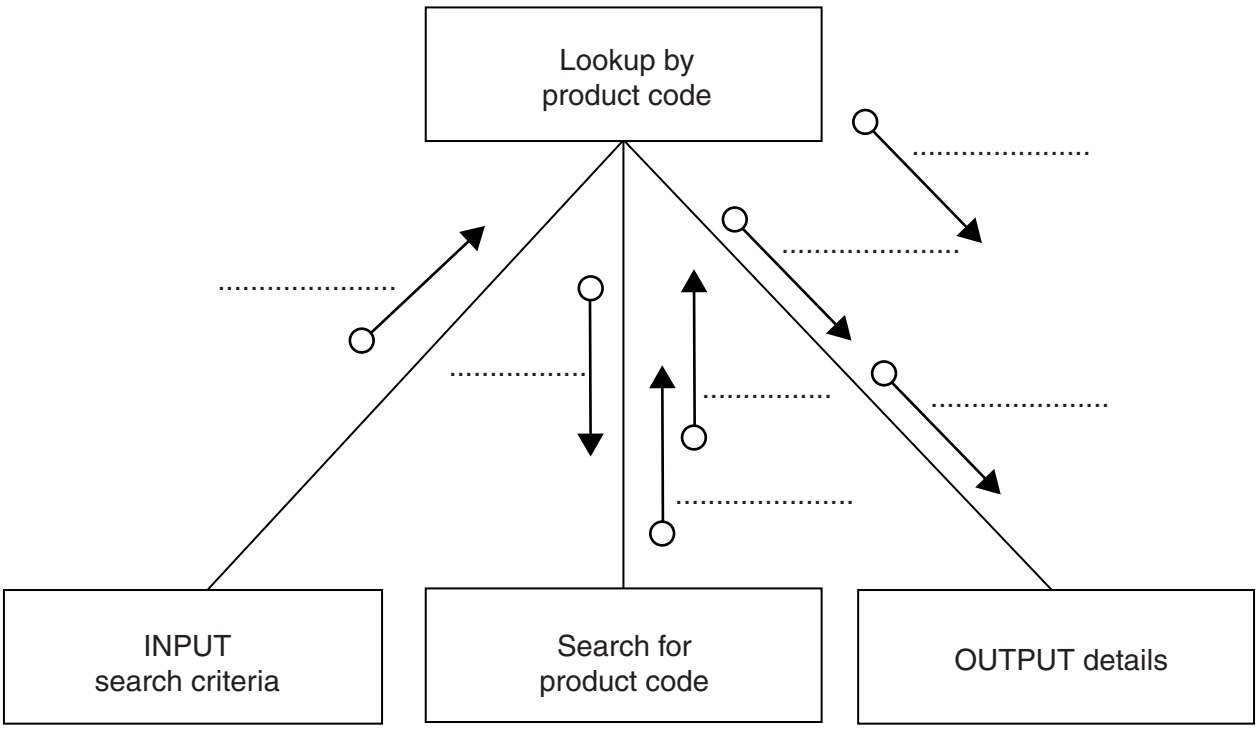
**(d)** To code the 'Search by product code' procedure, Ahmed draws a structure c… different stages.

The procedure uses the variables shown in the identifier table.

| Identifier | Data type | Description |
|---|---|---|
| `SearchCode` | `STRING` | Product code input by the user |
| `ThisIndex` | `INTEGER` | Array index position for the corresponding product |
| `ThisDescription` | `STRING` | Product description found |
| `ThisRetailPrice` | `REAL` | Product retail price found |

You can assume that before the procedure is run, all the product data is read from file `PRODUCTS` and then stored in three 1D arrays as described in **part (c)(i)**.

Label the structure chart to show the input(s) and output(s).




[4]

**(e)** A first attempt was made at writing the 'Search for product code' module.
Ahmed designs this as a function `ProductCodeSearch`.

The function returns an integer value as follows:

- if the product code is found, it returns the index position of the 1D array `PCode` be searched
- if the product code is not found, the function returns -1

Write **program code** for function `ProductCodeSearch`.

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 .....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[6]

# QUESTION 4.

**2** You will need to refer to the list of pseudocode string-handling functions in the **Appendix**.

**(a)** Give the value of the variables `x`, `y` and `z` for the following sequence of statements.

**(i)**
```
x ← ONECHAR("Barcelona", 5)
```
x ..... [1]

**(ii)**
```
y ← ONECHAR("Cool", 1) & ONECHAR("Ball", 2) & "t-food"
```
y ..... [1]

**(iii)**
```
Temp1 ← "13"
Temp2 ← ONECHAR("One-2-One", 5)
z ← TONUM(Temp2 & Temp1)
```
z ..... [1]

A computer program is to simulate the reading and processing of a string of characters from an input device.

The character string consists of:

- a number of digit characters
- one or more <*> characters, each used as a separator
- a final <#> character.

A typical input character sequence, stored as `InputString` is:

```
13*156*9*86*1463*18*#
```

Study this pseudocode.

```
01  DECLARE Numbers ARRAY [1:100] OF INTEGER
02  DECLARE InputString      : STRING
03  DECLARE NextChar         : CHAR
04  DECLARE NextNumberString : STRING
05  DECLARE i                : INTEGER    // Numbers array index
06  DECLARE j                : INTEGER    // InputString index
07
08  OUTPUT "String ... "
09  INPUT InputString
10  j ← 1
11  NextChar ← ONECHAR(InputString, j)
12
13  i ← 1
14  WHILE NextChar <> '#'
15    NextNumberString = ""
16    WHILE NextChar <> '*'
17      NextNumberString ← NextNumberString & NextChar
18      j ← j + 1
19      NextChar ← ONECHAR(InputString, j)
20    ENDWHILE
21
22    // store the next integer to the array
23    Numbers[i] ← TONUM(NextNumberString)
24    i ← i + 1
25    j ← j + 1
26    NextChar ← ONECHAR(InputString, j)
27  ENDWHILE
28
29  CALL DisplayArray()
```

**(b)** Write the line number for:

**(i)** A statement which declares a global variable used to store a single character. ..... [1]

**(ii)** A statement which runs code written as a procedure. ..... [1]

**(iii)** A statement which indicates the start of a 'pre-condition' loop. ..... [1]

**(iv)** A statement which increments a variable. ..... [1]

**(c)** Copy the condition which is used to control the inner loop.

..... [1]

**(d) (i)** Complete the trace table below for the given pseudocode as far as line 2

The input string is: `23*731*5*#`

| i | j | NextChar | NextNumberString | Numbers | | |
|---|---|---|---|---|---|---|
| | | | | 1 | 2 | 3 |
| 1 | 1 | '2' | | | | |
| | | | "" | | | |
| | | | "2" | | | |
| | 2 | '3' | "23" | | | |
| | 3 | '*' | | 23 | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |

[5]

**(ii)** Explain what this algorithm does.

.....

.....

.....

..... [2]

# QUESTION 5.

**2** One of the security features of a multi-user computer system is a user login pr
must complete this successfully before they can access the resources of the systen

As part of the login process the user enters their user ID followed by a password. The sys
compares the password entered with the password held in a file.

**(a)** The steps involved in the login process are described as follows:

- User enters their ID and password.
- Validation checks:
  - Compare user ID with data from the file.
  - Indicate whether or not the user ID was found.
  - If user ID found, check whether passwords match.

The description above is not detailed enough to allow a program to be written. The validation checks must be expressed as a more detailed algorithm.

Give the name of the process of increasing the level of detail of the algorithm.

.....[1]

**(b)** An identifier table is created as the algorithm is developed. A section of the table is shown. Complete the table.

| Identifier | Data Type | Description |
|---|---|---|
| `UserIDInput` | ..... | Stores the user ID entered |
| `PasswordInput` | ..... | ..... ..... |
| `UserIDFound` | ..... | ..... ..... |
| `PasswordValid` | ..... | ..... ..... |

[5]

**(c)** The validation checks described in **part (a)** are to be used as the basis for p

Use **structured English** to write a more detailed algorithm.

You should:

- use the identifiers given in the previous table
- assume that the password file, `password.txt`, is organised as a simple text file. The user ID and password are stored together, one entry per line.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[8]

# QUESTION 6.

**2** A multi-user computer system maintains a text file containing the ID and preferre
user.
User IDs are unique. Preferred names may be repeated.

**(a)** Stepwise refinement is to be applied to the following three steps.

After a user logs in, a welcome message is produced as follows:

1. Search for the user ID in the file.
2. Read the preferred name from the file.
3. Output the welcome message.

Describe the goal of **stepwise refinement**.

.....

.....

.....

..... [2]

**(b)** An initial identifier table is created as part of the stepwise refinement. A section of the table is shown. Complete this table.

| Identifier | Data type | Description |
|---|---|---|
| `SearchUserID` | | Stores the user ID entered |
| `FileUserID` | | ..... ..... |
| `FilePreferredName` | | ..... ..... |
| `IDFoundFlag` | | ..... ..... |

[5]

**(c)** The file may not contain an entry for every user. If the user ID is not found, "Guest" should be used as the preferred name.

Use **structured English** to write a more detailed algorithm for the welcome message described in **part (a)**.

Use the identifiers given in the previous table.

Assume that the file, `UserNames.txt`, is organised as a simple text file. The user ID and preferred name are held together in unencrypted form, one entry per line.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [8]

# QUESTION 7.

**2** The following pseudocode represents a simple algorithm.

```
DECLARE NumberFound, Remainder, Number : INTEGER
DECLARE StartNumber, EndNumber, Divisor : INTEGER

INPUT StartNumber
INPUT EndNumber
INPUT Divisor
NumberFound ← 0

FOR Number ← StartNumber TO EndNumber
   Remainder ← MODULUS(Number, Divisor)
   IF Remainder = 0
      THEN
         OUTPUT Number
         NumberFound ← NumberFound + 1
   ENDIF
ENDFOR
OUTPUT "Count: " & NumberFound
```

For the built-in functions list, refer to the **Appendix** on page 14.

**(a)** Complete the following trace table.

| StartNumber | EndNumber | Divisor | NumberFound | Number | Remainder | Output |
|---|---|---|---|---|---|---|
| 11 | 13 | 2 | 0 | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |

[3]

**(b)** Describe the purpose of this algorithm.

.....

.....

.....

.....

.....

.....[3]

**(c)** Draw a program flowchart to represent this algorithm.

Variable declarations are **not** required in program flowcharts.

[10]

# QUESTION 8.

**2** A 1D array, `ClassName`, of type `STRING` contains 100 elements.

The following pseudocode represents a simple algorithm to process the array.

```
DECLARE SearchValue : STRING
DECLARE FoundFlag : BOOLEAN
DECLARE Index : INTEGER

INPUT SearchValue
FoundFlag ← FALSE
Index ← 1

WHILE Index < 101 AND FoundFlag = False
   IF ClassName[Index] = SearchValue
      THEN
         OUTPUT Index
         FoundFlag ← TRUE
   ENDIF
   Index ← Index + 1
ENDWHILE

IF FoundFlag = FALSE
   THEN
      OUTPUT "Not found"
ENDIF
```

**(a)** Describe the purpose of the algorithm.

.....

.....

.....

.....

.....

..... [2]

**(b)** Draw a program flowchart to represent this algorithm.

Note that variable declarations are not required in program flowcharts.

[9]

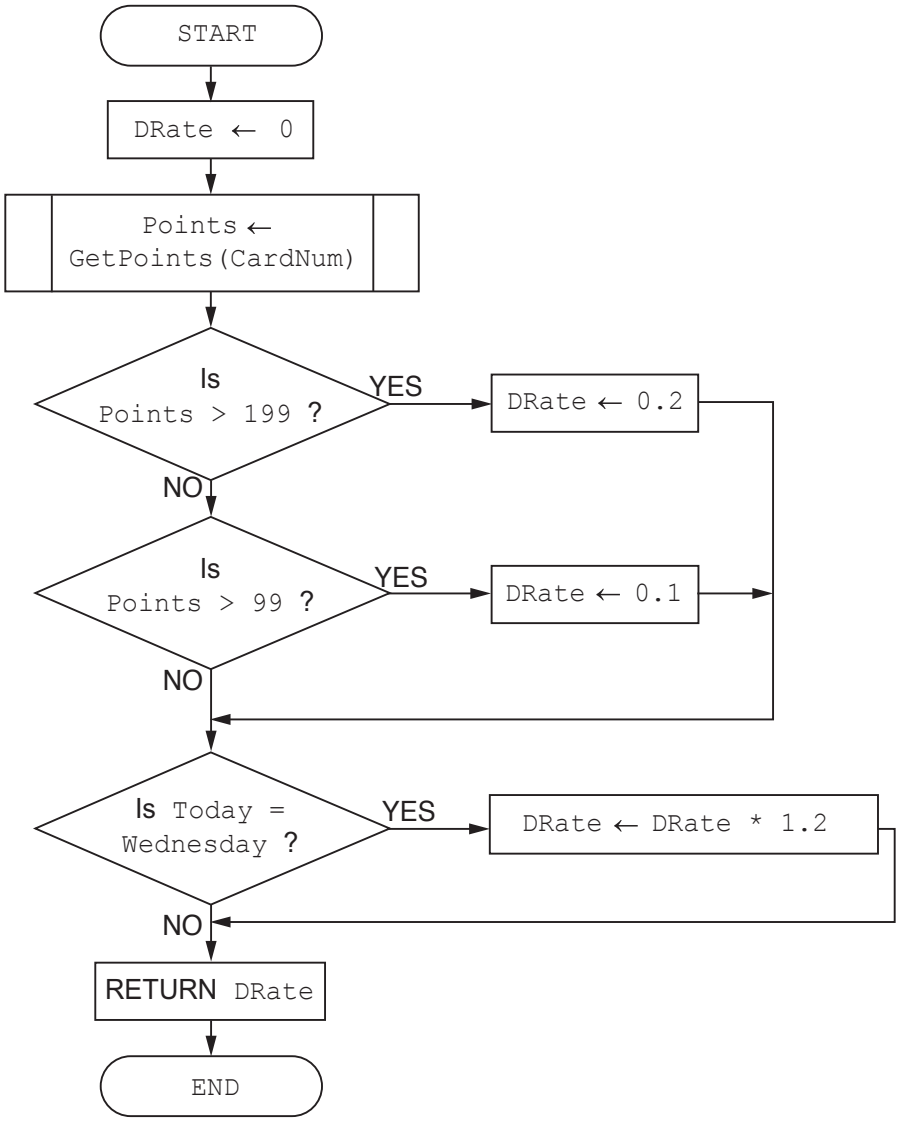
# QUESTION 9.

**2** Shop customers have a discount card with a unique card number. Customers co
they have bought items. The more points they have, the bigger the discount. If th
Wednesday, their discount is increased by 20%.

The function `GetDiscountRate()` takes a card number as a parameter and returns the dis
rate for a customer based on the number of points they have collected. A flowchart for the funct
is shown.

The function uses the following variables and functions.

| Identifier | Data type | Description |
|---|---|---|
| `DRate` | `REAL` | The discount rate |
| `CardNum` | `STRING` | The unique customer card number |
| `Points` | `INTEGER` | The number of points collected |
| `GetPoints()` | `FUNCTION` | Takes the card number as a parameter and returns the number of points already collected |
| `Today()` | `FUNCTION` | Returns the day number: 1 for Monday, 2 for Tuesday etc. |



**(a)** Write **pseudocode** to implement the `GetDiscountRate()` function.

Your solution should follow the flowchart for the function as closely as possibl
declarations should be included.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[8]

**(b)** A programmer writes the function `GetDiscountRate()` in a high-level lang

**(i)** A run-time error could occur when the function is used.

Name **and** describe **one** other type of error that the function could contain.

Name .....

Description .....

.....

.....

[2]

**(ii)** Function `GetPoints()` has not been written yet.

Name **and** describe a strategy that can be used to test `GetDiscountRate()` before the `GetPoints()` function has been written.

Name .....

Description .....

.....

.....

[2]

**(c)** There are different ways to minimise the risk of errors when writing programs, such as the use of constants and library routines.

**(i)** Identify **two** values that could be replaced by constants in the function `GetDiscountRate()`.

.....

.....[1]

**(ii)** Write **pseudocode** to declare **one** of the constants you have given in **part (c)(i)**.

.....[2]

**(iii)** Explain how the use of constants helps to minimise programming errors.

.....

.....

.....

.....[2]

**(iv)** Give a reason why the use of library routines helps to minimise the ris
writing a program.

.....

.....

**(v)** Constants and library routines help to minimise the risk of errors.

Name another way that you can minimise the risk of errors when writing a program. Explain how this helps.

Name .....

Explanation .....

.....

.....

[2]

# QUESTION 10.

3 **(a)** Programming is sometimes referred to as a **transferable skill**.

You are asked to work on a program written in a language you are not familiar wit

Explain how **transferable skills** would help you work on the program.

.....

.....

.....

.....[2]

**(b)** Stepwise refinement is often used in the development of an algorithm.

Describe **stepwise refinement**.

.....

.....

.....

.....[2]

**(c)** A program needs to search through 1000 elements of an unsorted array to find a given value.

The program will output:

- either the position in the array of the value
- or the message “Not Found”

Outline the steps the program needs to follow.

Do **not** write pseudocode or program code.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[4]

**Question 4 begins on the next page.**

# QUESTION 11.

**3** **(a)** A student is developing an algorithm to search through a 1D array of 100 element of the array, `Result`, contains a `REAL` value.

The algorithm will output:

- the average value of all the elements
- the number of elements with a value of zero.

The structured English description of the algorithm is:

1. SET Total value to 0
2. SET Zero count to 0
3. SELECT the first element
4. ADD value of element to Total value
5. IF element value is 0 then INCREMENT Zero count
6. REPEAT from step 4 for next element, until element is last element
7. SET Average to Total / 100
8. OUTPUT a suitable message and Average
9. OUTPUT a suitable message and Zero count

Write **pseudocode** for this algorithm.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [7]

**(b)** The student decides to change the algorithm and implement it as a procedure, which will be called with three parameters.

```
ScanArray(AverageValue, ZeroCount, ArrayName)
```

`ScanArray()` will modify the first two parameters so that the new values are available to the calling program or module.

Write the **pseudocode** procedure header for `ScanArray()`.

.....

.....

.....

..... [4]

**Question 4 begins on the next page.**

# QUESTION 12.

**5** A student is learning about text files. She wants to write a program to count the number of lines in a file.

**(a)** Use **structured English** to describe an algorithm she could use.

.....

.....

.....

.....

.....

..... [3]

**(b)** A procedure, `CountLines()`, is being written to count the number of lines in a file. The procedure will:

- take a filename as a string parameter
- count the number of lines in the file
- output a single suitable message that includes the total number of lines.

Write **pseudocode** for the procedure `CountLines()`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [6]

# QUESTION 13.

**5** Nigel is learning about string handling. He wants to write code to count the numb
given string. A word is defined as a sequence of alphabetic characters that is separa
more space characters.

His first attempt at writing an algorithm in pseudocode is as follows:

```
PROCEDURE CountWords(Message : STRING)

   DECLARE NumWords : INTEGER
   DECLARE Index : INTEGER
   CONSTANT Space = ' '

   NumWords ← 0

   FOR Index ← 1 TO LENGTH(Message)
      IF MID(Message, Index, 1) = Space
         THEN
            NumWords ← NumWords + 1
      ENDIF
   ENDFOR

   OUTPUT "Number of words : " , NumWords

ENDPROCEDURE
```

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

His first attempt is incorrect. He will use white-box testing to help him to identify the problem.

**(a) (i)** State the purpose of white-box testing.

.....

..... [1]

**(ii)** Dry running the code is often used in white-box testing. In this method, the programmer records the values of variables as they change.

Identify what the programmer would normally use to record the changes.

..... [1]

**(b) (i)** Write a test string containing two words that gives the output:

```
Number of words : 2
```

Use the symbol `'∇'` to represent each space character in your test string.

Explain why the algorithm gives the output shown above.

String .....

Explanation .....

.....

.....

.....

.....

[3]

**(ii)** Nigel tested the procedure with the strings:

String 1: `"Red∇and∇Yellow"`
String 2: `"Green∇∇and∇∇Pink∇"`

Give the output that is produced for each of the strings.

Describe the changes that would need to be made to the algorithm to give the correct output in each case.

Do **not** write pseudocode **or** program code.

String 1 .....

Description .....

.....

.....

.....

String 2 .....

Description .....

.....

.....

.....

[6]

# QUESTION 14.

**4** A student is developing a program to count how many times each character of the occurs in a given string. Upper case and lower case characters will be counted as t string may contain non-alphabetic characters, which should be ignored.

The program will:

- check each character in the string to count how many times each alphabetic character occurs
- store the count for each alphabetic character in a 1D array
- output each count together with the corresponding character.

**(a)** The student has written a structured English description of the algorithm:

1. START at the beginning of the string
2. SELECT a character from the string
3. CONVERT the character to upper case
4. CHECK whether the character is alphabetic and INCREMENT as required.
5. REPEAT from step 2 until last character has been checked
6. OUTPUT a suitable message giving the count of each alphabetic character

Step 4 above is not described in sufficient detail.

The student decides to apply a process to increase the level of detail given in step 4.

State the name of the process **and** use this process to write step 4 in more detail. Use **structured English** for your answer.

Process .....

Structured English .....

.....

.....

.....

.....

.....

[4]

**(b)** Write **pseudocode** to implement the program.

You should note the following:

- `InString` contains the string to be checked. It has been assigned a value.
- The elements of the array `Result` have all been initialised to zero.
- The ASCII value of letter 'A' is 65.

You should assume the following lines of pseudocode have been written:

```
DECLARE InString : STRING
DECLARE Result : ARRAY [1:26] OF INTEGER
```

Declare any further variables you use. Do **not** implement the code as a subroutine.

Refer to the **Appendix** on pages 16–17 for the list of built-in functions and operators.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [7]

# QUESTION 15.

**1** Study the following pseudocode.

```
PROCEDURE FillTank()

   DECLARE Tries : INTEGER
   DECLARE Full : BOOLEAN

   Tries ← 1

   Full ← ReadSensor("F1")

   IF NOT Full
      THEN
         WHILE NOT Full AND Tries < 4
            CALL TopUp()
            Full ← ReadSensor("F1")
            Tries ← Tries + 1
         ENDWHILE
         IF Tries > 3
            THEN
               OUTPUT "Too many attempts"
            ELSE
               OUTPUT "Tank now full"
         ENDIF
      ELSE
         OUTPUT "Already full"
   ENDIF

ENDPROCEDURE
```

**(a) (i)** The pseudocode includes features that make it easier to read and understand.

State **three** such features.

Feature 1 .....

Feature 2 .....

Feature 3 .....

[3]

**(ii)** Draw a program flowchart to represent the algorithm implemented in the
Variable declarations are not required in program flowcharts.

[5]

**(b) (i)** Programming languages support different data types.

Complete the table by giving a suitable data type for each example value.

| Example value | Data type |
|---|---|
| `43` | |
| `TRUE` | |
| `-273.16` | |
| `"-273.16"` | |

[4]

**(ii)** Evaluate each expression in the following table.

If an expression is invalid then write 'ERROR'.

Refer to the **Appendix** on page 18 for the list of built-in functions and operators.

| Expression | Evaluates to |
|---|---|
| `RIGHT("Stop", 3) & LEFT("ich", 2)` | |
| `MID(NUM_TO_STRING(2019), 3, 1)` | |
| `INT(NUM_TO_STRING(-273.16))` | |
| `INT(13/2)` | |

[4]

# QUESTION 16.

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

```
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. Each should test a different rule.

Justify your choices.

Value .....

Justification .....

.....

.....

Value .....

Justification .....

.....