

QUESTION 6.



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]



(b) The programmer writes a program from the following pseudocode design.

```

x ← 0
FOR DayNumber ← 1 TO 7
    IF SalesX[DayNumber] + SalesY[DayNumber] >= 10
        THEN
            x ← x + 1
            OUTPUT SalesDate[DayNumber]
        ENDIF
    ENDFOR
OUTPUT x
    
```

(i) Trace the execution of this pseudocode by completing the trace table below.

x	DayNumber	OUTPUT
0		

[4]

(ii) Describe, in detail, what this algorithm does.

.....

.....

.....

.....[3]



- (c) The company wants a program to output the total monthly sales for one websites.

The programmer codes a function with the following function header:

```
FUNCTION MonthlyWebSiteSales(ThisMonth : INTEGER, ThisSite : CHAR)
                                RETURNS INTEGER
```

The function returns the total number of bicycles sold for the given month and website.

The function will use the following:

Identifier	Data type	Description
ThisMonth	INTEGER	Represents the month number e.g. 4 represents April
ThisSite	CHAR	Coded as: <ul style="list-style-type: none"> • X for website X • Y for Website Y

- (i) Give the number of parameters of this function.[1]
- (ii) Some of the following function calls may be invalid.

Mark each call with:

- a tick (✓), for a valid call
- a cross (✗), for an invalid call

For any function calls which are invalid, explain why.

Function call	Tick (✓) /cross (✗)	Explanation (if invalid)
MonthlyWebSiteSales(1, "Y")		
MonthlyWebSiteSales(11, 'X', 'Y')		
MonthlyWebSiteSales(12, 'X')		

[3]

- (d) The company decides to offer a discount on selected dates. A program is written to generate a text file, DISCOUNT_DATES, containing the dates on which a discount is offered.

The program creates a text file, DISCOUNT_DATES (with data as shown), for a number of consecutive dates.

03/06/2015 TRUE
04/06/2015 FALSE
05/06/2015 FALSE
06/06/2015 FALSE
07/06/2015 FALSE
08/06/2015 FALSE
09/06/2015 FALSE
10/06/2015 TRUE
11/06/2015 FALSE

01/07/2015 FALSE

Each date and discount indicator is separated by a single <Space> character.

The discount indicators are:

- FALSE – indicates a date on which no discount is offered
- TRUE – indicates a date on which a discount is offered

A programming language has the built-in function CONCAT defined as follows:

```
CONCAT(String1 : STRING, String2 : STRING [, String3 : STRING] )
                                                    RETURNS STRING
For example:   CONCAT("San", "Francisco") returns "SanFrancisco"
               CONCAT("New", "York", "City") returns "NewYorkCity"
```

The use of the square brackets indicates that the parameter is optional.



The following incomplete pseudocode creates the text file DISCOUNT_DATES.

Complete the pseudocode.

```
OPENFILE "DISCOUNT_DATES" FOR .....  
INPUT .....  
WHILE NextDate <>"XXX"  
    INPUT Discount  
    ..... = CONCAT(NextDate, " ", Discount)  
    WRITEFILE "DISCOUNT_DATES", NextLine  
    INPUT NextDate  
    .....  
OUTPUT "File now created"  
CLOSEFILE
```

[4]



Question 5(e) continues on page 18.



(e) The `DISCOUNT_DATES` text file is successfully created.

The company now wants a program to:

- key in a date entered by the user
- search the text file for this date
- if found, output one of the following messages:
 - “No discount on this date”
 - “This is a discount date”
- if not found, output “Date not found”

(i) Add to the identifier table to show the variables you need for this new program.

Identifier	Data type	Description
<code>DISCOUNT_DATES</code>	<code>FILE</code>	Text file to be used

[3]

QUESTION 7.



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 the built-in functions below:

ONECHAR(ThisString : STRING, Position : INTEGER) RETURNS CHAR
 returns the single character at position Position (counting from the start of the string with value 1) 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("BRAZIL") returns 6.

CHR(ThisInteger : INTEGER) RETURNS CHAR
 returns the character with ASCII code ThisInteger.
 For example: CHR(65) returns character 'A'.

ASC(ThisCharacter : CHAR) RETURNS INTEGER
 returns the ASCII value for character ThisCharacter.
 For example: ASC('A') returns 65.

(a) Show the values stored by variables A, B, C and D.

The & operator is used to concatenate two strings.

Num1 ← 15	
A ← CHR(67) & CHR(65) & CHR(84)	(i) A [1]
B ← ASC('P') - ASC('F') + 3	(ii) B [1]
C ← ASC(ONECHAR("BISCUITS", 3))	(iii) C [1]
D ← CHARACTERCOUNT("New York City") + 2	(iv) D [1]

QUESTION 8.



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.

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

- (i) A [1]
- (ii) B [1]
- (iii) C [1]
- (iv) D [1]

QUESTION 9.



5 Toni has a large collection of jazz CDs that are stored in different places. She where the CDs are stored. She decides to write a program to do this.

The program must store the data in a file, `MyMusic`.

(a) (i) Why is a file needed?

.....
.....[1]

(ii) `MyMusic` is a text file with the data for each CD as one line of text.

Data for a typical CD are:

Title: Kind of Green
Artist: Miles Coltrane
Location: Rack1-5

The line will be formed by concatenating the three data items.

For the example above, the line stored will be:

`Kind of GreenMiles ColtraneRack1-5`

Describe a problem that might occur when organising the data in this way.

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

Describe a possible solution.

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

[4]



(b) Toni must input the data into the file for all of her CDs.

A procedure, `InputData`, is needed to do this.

Toni designs the procedure and chooses the following identifiers:

Identifier	Data type
<code>CDTitle</code>	STRING
<code>CDArtist</code>	STRING
<code>CDLocation</code>	STRING

The procedure repeatedly performs the following steps:

- input a CD title (A rogue value of “##” is to be used to end the input)
- input the artist
- input the location
- create the text line
- write the text line to the file

When the rogue value is encountered the file is closed.

QUESTION 10.

11



- 5 Claudia stores her large collection of music CDs in different places. Claudia wants she stores each CD. She decides to write a program to do this.

Data items for a typical CD are:

Title: Kind of Green
Artist: Miles Coltrane
Location: Rack3-23

The data is to be stored in a text file, `MyMusic`. Each line of the text file will be a string, formed by concatenating the three data items.

Before concatenation, the title and artist will each be made into a fixed-length string of 40 characters. Space characters may need to be added to each data item.

The location is always 8 characters long.

- (a) (i) Explain the benefit of making the stored data into fixed-length strings.

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

State a drawback of this file design.

.....
.....

[3]



- (ii) When Claudia buys a new CD, the CD data must be added to the existing file. She has written a procedure in pseudocode. This has the following statements:

```

OPENFILE "MyMusic" FOR WRITE
WRITEFILE "MyMusic", OutputString
CLOSEFILE "MyMusic"
    
```

There is a problem with the logic of this pseudocode.

State the problem.

.....

.....

Identify the effect it will have if the final code is implemented in this way.

.....

.....

Give a possible solution.

.....

.....

[3]

- (b) Claudia needs to output a list of all the CDs in a particular location.

She designs a procedure, `OutputLocationList`, to do this. She also chooses the following identifiers:

Identifier	Data type
CDTitle	STRING
CDArtist	STRING
CDLocation	STRING

The procedure will:

- prompt for the name of the location
- input the location (such as "Rack3-23")
- search the file for all CDs at this location
- output the title and artist of each CD found
- output the total number of CDs found at that location (such as "17 CDs found")



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



- (i) The first operation of the program is to read all the product data held in the file `PRODUCTS` 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]



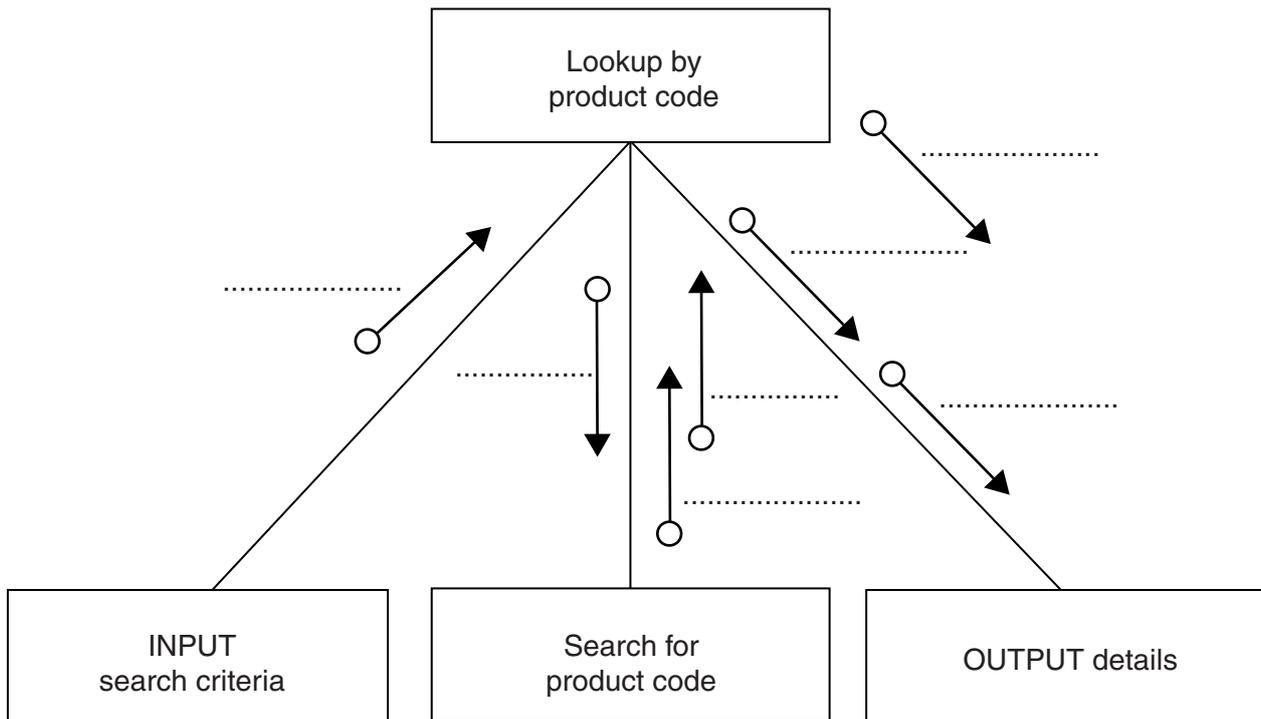
(d) To code the 'Search by product code' procedure, Ahmed draws a structure chart with different stages.

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]

QUESTION 12.



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

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.

```
01 INPUT GameNumber
02 INPUT PlayerNumber
03 INPUT PlayerGameScore
04 PlayerScore[GameNumber, PlayerNumber] ← PlayerGameScore
```

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

..... [2]



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

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 Total150 ← 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             Total150 ← Total150 + GameIndex
12           ENDIF
13       ENDIF
14   ENDFOR
15 ENDFOR
16 OUTPUT Total150
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]



- (b) The value of `UserID` should be unique for each user but a problem has occurred and repeated `UserID` values may have been issued.

The array is sorted by `UserID`, so any repeated `UserID` values will appear in consecutive array elements.

A procedure, `FindRepeats` is required.

This will:

- compare each element with the previous element and output the `UserID` and `UserName` if the `UserID` is repeated
- output the total number of `UserIDs` that are repeated.

For example, the `UserNameArray` contains the following entries.

Array element	Comment
<code>123456</code>	
<code>122222Jim Moriarty</code>	
<code>123456</code>	
<code>123456Fred Smith</code>	
<code>123456Eric Sykes</code>	Repeated User ID
<code>123456Kevin Turvey</code>	Repeated User ID
<code>222244</code>	
<code>222244Alice Chan</code>	
<code>222244Myra Singh</code>	Repeated User ID
<code>333333</code>	
<code>333333Yasmin Halim</code>	
<code>444444</code>	

For this example, the output is:

```
123456Eric Sykes
123456Kevin Turvey
222244Myra Singh
There are 3 repeated UserIDs
```

If no repeated `UserIDs` are found, the output is:

```
The array contains no repeated UserIDs
```




(c) (i) The `FindRepeats` procedure forms part of a program.

Name **three** stages in a program development cycle.

- 1
- 2
- 3 [3]

(ii) The program containing `FindRepeats` will be created using an IDE.

State what is meant by IDE.

-
- [1]

(iii) Name **two** features provided by an IDE that assist in the program development cycle.

- 1
-
- 2
- [2]

(iv) The procedure, `FindRepeats`, is written assuming there are 100 elements in `UserNameArray`.

In the main program, the global array, `UserNameArray`, has been declared with only 50 elements.

State the type of error this will cause.

- [1]

QUESTION 16.



- 4 Programming languages provide built-in functions to generate random numbers. To be truly random, the frequency of each number generated should be the same.

You are required to write program code to test the random number generator of your language.

The test should:

- generate a given number of random numbers between 1 and 10 inclusive
- keep a count of the number of times each number is generated
- calculate the expected frequency of each number 1 to 10
- output the actual frequency of each number 1 to 10
- output the difference between the actual frequency and the expected frequency.

The program code should be written as a procedure. In pseudocode, the procedure heading will be:

```
PROCEDURE TestRandom(Repetitions AS INTEGER)
```

The parameter, `Repetitions`, contains a value representing the total number of random numbers that should be generated.

The following example shows the expected output for the procedure call, `TestRandom(200)`.

The expected frequency is 20.

Number	Frequency	Difference
1	17	-3
2	21	1
3	12	-8
4	28	8
5	20	0
6	19	-1
7	21	1
8	16	-4
9	24	4
10	22	2



(b) Name **three** features of a typical IDE that would help a programmer to debug

Explain how each of these could be used in the debugging of the `TestRandom` from **part (a)**.

Feature 1

Explanation

.....

.....

.....

Feature 2

Explanation

.....

.....

.....

Feature 3

Explanation

.....

.....

.....

[6]

(c) The procedure is developed and run using the call `TestRandom(200)`. No system errors are produced.

To ensure that the procedure works correctly, you need to check the output.

Describe **two** checks you should make to suggest the program works correctly.

1

.....

.....

2

.....

.....

[2]



(c) The function `ScanArray()` is one of a number of sub-tasks within a program.

Name the process that involves the splitting of a problem into sub-tasks **and** list two advantages of this approach.

Name

Advantage 1

.....

Advantage 2

.....

[3]

(d) `ResultArray` is a 2D array of type `STRING`. It represents a table containing 100 rows and 2 columns.

Write **program code** to declare `ResultArray` **and** set all elements to the value `'*'`.

Programming language

Program code

.....

.....

.....

.....

.....

.....

.....

.....

[3]



Question 5 begins on the next page.

QUESTION 18.



- 6 Nadine is developing a program to store the ID and preferred name for each student. For example, student Pradeep uses the preferred name “Prad”.

The program will:

1. prompt and input a valid user ID and a preferred name
2. write the user ID and preferred name to one of two files
3. allow the user to end the program or repeat from step 1.

The program will consist of three separate modules. Each module will be implemented using either a procedure or a function.

Nadine has defined the modules as follows:

Module	Description
<code>TopLevel()</code>	<ul style="list-style-type: none">• Call <code>GetInfo()</code> to obtain a string containing a valid user ID and a preferred name• Call <code>WriteInfo()</code> to write the string to either <code>File1.txt</code> or <code>File2.txt</code> depending on the first character of the user ID as follows:<ul style="list-style-type: none">○ 'A' to 'M': writes to <code>File1.txt</code>○ 'N' to 'Z': writes to <code>File2.txt</code>For example, a string with a user ID of "G1234" writes to <code>File1.txt</code>• End the program if the file write was unsuccessful• Input (Y/N) to either repeat for the next user ID or to end the program
<code>GetInfo()</code>	<ul style="list-style-type: none">• Input a user ID and repeat until the user ID is valid• Input a preferred name. This will be an empty string if no preferred name is input.• Concatenate the user ID and preferred name using a '*' character as a separator and return this string
<code>WriteInfo()</code>	<ul style="list-style-type: none">• Open the file• Append the concatenated string to the file• Close the file• Return a Boolean value:<ul style="list-style-type: none">○ TRUE if the file write was successful○ FALSE if the file write failed, for example, if the disk was full

A valid user ID:

- is five characters in length
- has a single upper case alphabetic character followed by four numeric characters, for example “G1234”.

Nadine has decided that global variables and nested modules must not be used.

Nadine wants all inputs to have suitable prompts.



(c) Write **pseudocode** for the module declaration of `WriteInfo()`.

.....

.....

.....

..... [3]

Permission to reproduce items where third-party owned material protected by copyright is included has been sought and cleared where possible. Every reasonable effort has been made by the publisher (UCLES) to trace copyright holders, but if any items requiring clearance have unwittingly been included, the publisher will be pleased to make amends at the earliest possible opportunity.

To avoid the issue of disclosure of answer-related information to candidates, all copyright acknowledgements are reproduced online in the Cambridge Assessment International Education Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at www.cambridgeinternational.org after the live examination series.

Cambridge Assessment International Education is part of the Cambridge Assessment Group. Cambridge Assessment is the brand name of the University of Cambridge Local Examinations Syndicate (UCLES), which itself is a department of the University of Cambridge.

QUESTION 19.



- 6 A text file, `StudentContact.txt`, contains a list of names and telephone numbers of students in a school. Not all students in the school have provided a contact telephone number. Students who have not provided their name will not be in the file.

Each line of the file is stored as a string that contains a name and telephone number, separated by the asterisk character ('*') as follows:

`<Name>' * '<TelNumber>`, for example:

```
"Bill Smith*081234567"
```

A 1D array, `ClassList`, contains the names of students in a particular class. The array consists of 40 elements of string data type. You can assume that student names are unique. Unused elements contain the empty string "".

A program is to be written to produce a **new** text file, `ClassContact.txt`, containing student names and numbers for all students in a particular class.

For each name contained in the `ClassList` array, the program will:

- search the `StudentContact.txt` file
- copy the matching string into `ClassContact.txt` if the name is found
- write the name together with “*No number” into `ClassContact.txt` if the name is not found.

The program will be implemented as three modules. The description of these is as follows:

Module	Description
<code>ProcessArray()</code>	<ul style="list-style-type: none">• Check each element of the array:<ul style="list-style-type: none">○ Read the student name from the array○ Ignore unused elements○ Call <code>SearchFile()</code> with the student name○ If the student name is found, call <code>AddToFile()</code> to write the student details to the class file○ If the student name is not found, call <code>AddToFile()</code> to write a new string to the class file, formed as follows: <code><Name>*No number</code>• Return the number of students who have not provided a telephone number
<code>SearchFile()</code>	<ul style="list-style-type: none">• Search for a given student name at the start of each line in the file <code>StudentContact.txt</code>:<ul style="list-style-type: none">○ If the search string is found, return the text line from <code>StudentContact.txt</code>○ If the search string is not found, return an empty string
<code>AddToFile()</code>	<ul style="list-style-type: none">• Append the given string to a specified file, for example, <code>AddToFile(StringName, FileName)</code>

(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 20.



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

The format of the data held in the two files is identical. Each line of the file is stored as a string. Each line contains an account number, name and telephone number separated by the asterisk character. The format is: `'*'*'<Name>'*'*'<Telephone Number>` 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
<code>ClearArray()</code>	<ul style="list-style-type: none">• Initialise the global array <code>Duplicates</code>. Set all elements to the empty string.
<code>FindDuplicates()</code>	<ul style="list-style-type: none">• Read each line from the file <code>UserListAtoM.txt</code><ul style="list-style-type: none">◦ Check whether the account number appears in file <code>UserListNtoZ.txt</code> using <code>SearchFileNtoZ()</code>◦ 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.
<code>SearchFileNtoZ()</code>	<ul style="list-style-type: none">• Search for a given account number in file <code>UserListNtoZ.txt</code><ul style="list-style-type: none">◦ If found, return <code>TRUE</code>, otherwise return <code>FALSE</code>

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

.....
..... [1]



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





QUESTION 21.



- 6 A text file, `Library.txt`, stores information relating to a book collection. The pieces of information about each book on separate lines of the file, as follows:

```
Line n:      <Book Title>
Line n + 1:  <Author Name>
Line n + 2:  <ISBN>
Line n + 3:  <Location>
```

Information is stored as data strings.

Information relating to two books is shown:

File line	Data
100	"Learning Python"
101	"Brian Smith"
102	"978-14-56543-21-8"
103	"BD345"
104	"Surviving in the mountains"
105	"C T Snow"
106	"978-35-17635-43-9"
107	"ZX001"

- (a) (i) A function, `FindBooksBy()`, will search `Library.txt` for all books by a given author.

The function will store the `Book Title` and `Location` in the array `Result`, and will return a count of the number of books found.

Array `Result` is a global 2D array of type `STRING`. It has 100 rows and 2 columns.

Write **pseudocode** to declare the array `Result`.

.....
.....
..... [3]

- (ii) Function `FindBooksBy()` will:

- receive the `Author Name` as a parameter
- search `Library.txt` for matching entries
- store the `Book Title` and `Location` of matching entries in the `Result` array
- return an integer value giving the number of books by the author that were found.



.....

.....

.....

.....

.....

.....

..... [8]

(b) The function `FindBooksBy()` has already been called and has stored values in the array `Result`.

The procedure, `DisplayResults()`, will output the information from the array.

The procedure receives the following two parameters:

- a string containing the author name
- an integer value representing the number of books found

The output should be formatted as in the following example:

```
Books written by: Brian Smith

Title                Location
Learning Python      BD345
Arrays are not lists CZ562
Learning Java         CZ589

Number of titles found: 3
```

If no books by the author are found, the following should be output:

```
Search found no books by: Brian Smith
```

