Cambridge Assessment International Education

Cambridge International AS & A Level

CANDIDATE NAME									
CENTRE NUMBER						CANDIDATE NUMBER			

1790695172

COMPUTER SCIENCE **9618/23**

Paper 2 Fundamental Problem-solving and Programming Skills **October/November 2023**

2 hours

You must answer on the question paper.

You will need: Insert (enclosed)

INSTRUCTIONS

- Answer **all** questions.
- Use a black or dark blue pen.
- Write your name, centre number and candidate number in the boxes at the top of the page.
- Write your answer to each question in the space provided.
- Do **not** use an erasable pen or correction fluid.
- Do **not** write on any bar codes.
- You may use an HB pencil for any diagrams, graphs or rough working.
- Calculators must **not** be used in this paper.

INFORMATION

- The total mark for this paper is 75.
- The number of marks for each question or part question is shown in brackets [ ].
- No marks will be awarded for using brand names of software packages or hardware.
- The insert contains all the resources referred to in the questions.

This document has **20** pages.

Refer to the **insert** for the list of pseudocode functions and operators.

1 A program is being developed in pseudocode before being converted into a programming language.

(a) The following table shows four valid pseudocode assignment statements.

Complete the table by giving the data type that should be used to declare the variable underlined in each assignment statement.

Assignment statement	Data type
MyVar1 ← Total1 / Total2	REAL
MyVar2 ← 27/10/2023	DATE
MyVar3 ← "Sum1 / Sum2"	STRING
MyVar4 ← Result1 AND Result2	BOOLEAN

[4]

(b) Other variables in the program have example values as shown:

Variable	Value
Active	TRUE
Fraction	0.2
Code	"Ab12345"

Complete the table by evaluating each expression using the example values.

Expression	Evaluates to
Fraction >= 0.2 AND NOT Active	FALSE
INT((Fraction * 100) + 13.3)	33
STR_TO_NUM(MID(Code, 4, 2)) + 5	28
LENGTH("TRUE" & Code)	11

[4]

(c) The program makes use of complex statistical functions. The required functions are not built-in to the programming language and are too complicated for the programmer to write.

One solution would be to employ another programmer who has experience of writing these functions, as there is no time to train the existing programmer.

State **one other** way that these functions may be provided for inclusion in the program.

..... The use of a program library (routines) .....

..... [1]

(d) The hardware that runs the program is changed and the program needs to be modified so that it works with the new hardware.

Identify the type of maintenance that this represents **and** give **one other** reason why this type of maintenance may be needed.

Type .....**MP1** Type: Adaptive..

Reason **MP2** Reason: The (user) requirement(s) changes // to accommodate legislative changes..

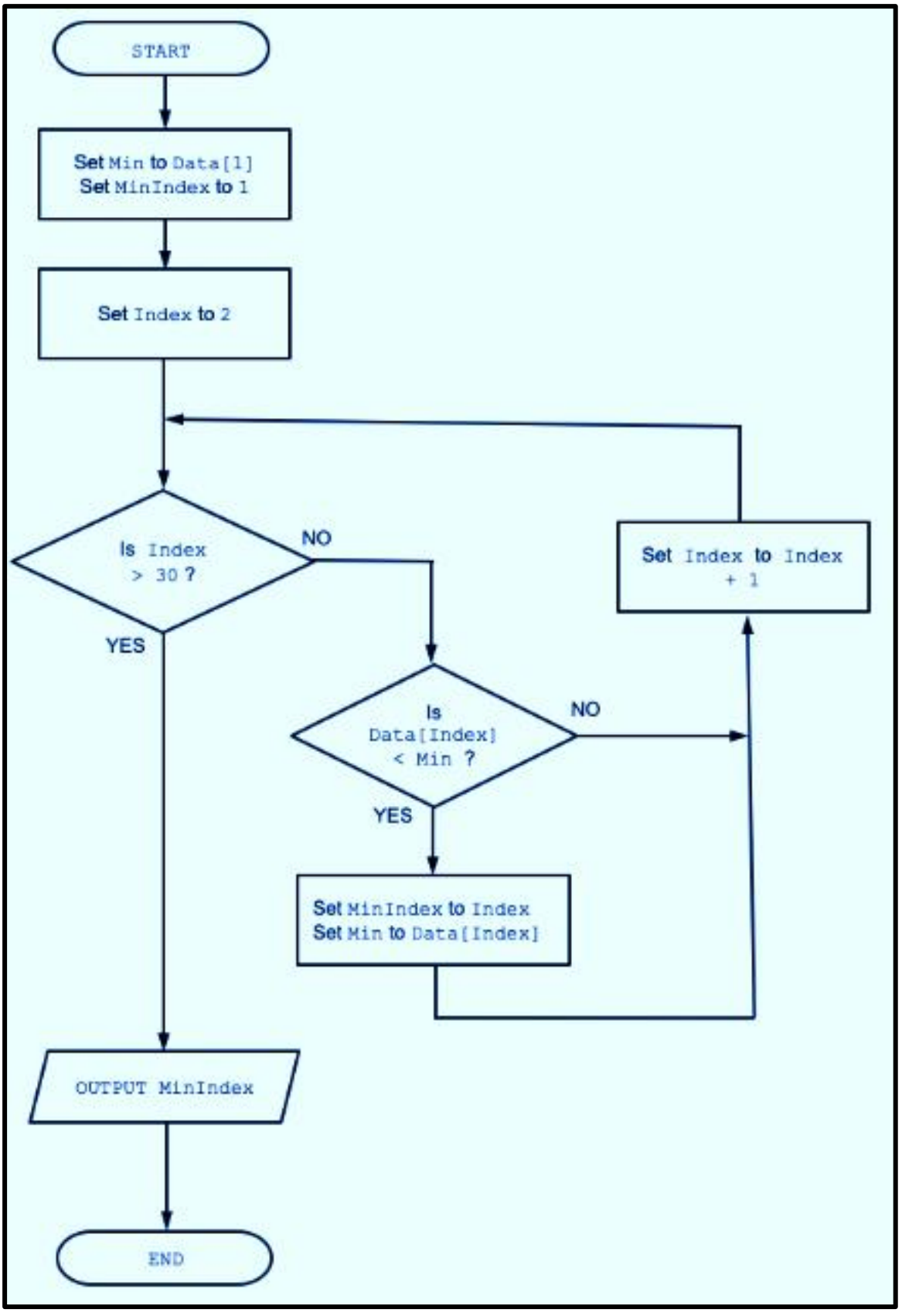
.....

[2]

2 `Data` is a 1D array of integers, containing 30 elements. All element values are unique.

(a) An algorithm will output the index of the element with the smallest value.

Draw a program flowchart to represent the algorithm.



[5]

(b) The 30 data values could have been stored in separate variables rather than in an array.

Explain the benefits of using an array when designing a solution to part **(a)**.

MP1 Simplifies the algorithm // easier to write / understand / test / debug
MP2 It is possible to iterate through the values // can use a loop // allows the storage of many values using a single identifier [2]

(c) The requirement changes. Array `Data` needs to hold 120 elements and each value may include a decimal place.

Write a pseudocode statement to declare the modified array.

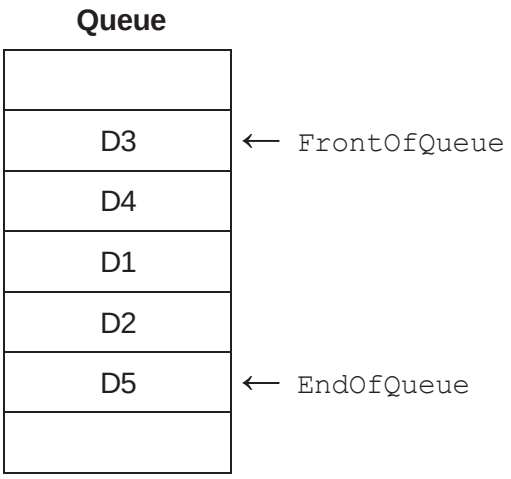
```
DECLARE Data : ARRAY[1:120] OF REAL
```

[2]

3 The diagram represents a queue Abstract Data Type (ADT).

The organisation of this queue may be summarised as follows:

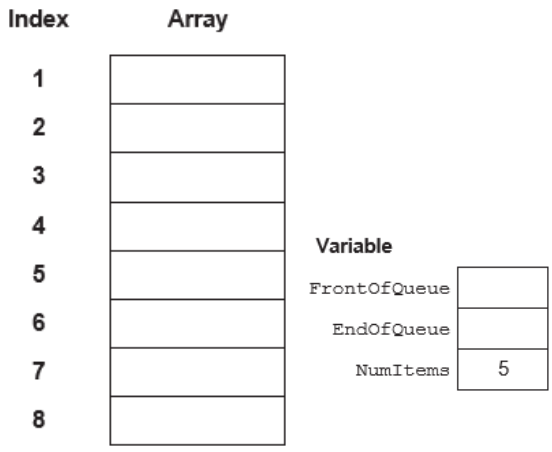
- The `FrontOfQueue` pointer points to the next data item to be removed.
- The `EndOfQueue` pointer points to the last data item added.

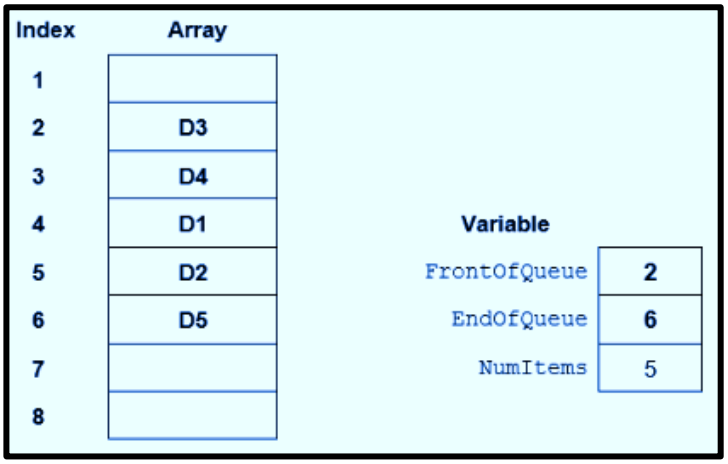


The queue is implemented using three variables and a 1D array of eight elements as shown. The variable `NumItems` stores the number of items in the queue.

The pointer variables store indices (index numbers) of the array.

(a) Complete the diagram to represent the state of the queue as shown above.





(b) A module `AddTo()` will add a value to the queue by manipulating the array and variables in part **(a)**.

The queue implementation is circular. When pointers reach the end of the queue, they will 'wrap around' to the beginning.

Before a value can be added to the queue, it is necessary to check the queue is not full.

The algorithm to add a value to the queue is expressed in six steps.

Complete the steps.

1. If `NumItems` . . . . . then jump to step 6.
2. Increment ..... .
3. If ..... then set `EndOfQueue` to ..... .
4. Increment ..... .
5. Set the ..... at the index stored in ..... to the ..... being added.
6. Stop.

[6]

MP1 If `NumItems` **is / = 8 // (queue) full** then jump to step 6
MP2 Increment **`EndOfQueue`**
MP3 If **`EndOfQueue`** **= 9** then set `EndOfQueue` to **1**
MP4 Increment **`NumItems`**

MP5 Set the **Element** at the index …
MP6 stored in **`EndOfQueue`** to **value/data/item** being added

4 A procedure `RandList()` will output a sequence of 25 random integers, where each integer is larger than the previous one.

(a) Write pseudocode for procedure `RandList()`.

```
PROCEDURE RandList()
   DECLARE Count, BaseNum, ThisNum : INTEGER
   CONSTANT StepVal = 10
   BaseNum ← 0

   FOR Count ← 1 TO 25
      ThisNum ← BaseNum + INT(RAND(StepVal))
      OUTPUT ThisNum
      BaseNum ← BaseNum + StepVal
   NEXT Count

ENDPROCEDURE
```

[6]

(b) Procedure `RandList()` is modified so that the random numbers are also written into a 1D array `Result`.

A new module is written to confirm that the numbers in the array are in ascending order.

This module contains an `IF` statement that performs a comparison between elements:

```
IF (Result[x + 1] = Result[x]) OR (Result[x] > Result[x + 1]) THEN
   Sequence ⟵ FALSE
ENDIF
```

Write a simplified version of the conditional clause.

..... **MP1** `Result[x + 1] <= Result[x]` .....

.....

ALTERNATIVE SOLUTION:

..... `Result[x] >= Result[x + 1]` ..... [1]

5 A global 1D array of integers contains four elements, which are assigned values as shown:

```
Mix[1] ← 4
Mix[2] ← 2
Mix[3] ← 3
Mix[4] ← 5
```

A procedure `Process()` manipulates the values in the array.

The procedure is written in pseudocode as follows:

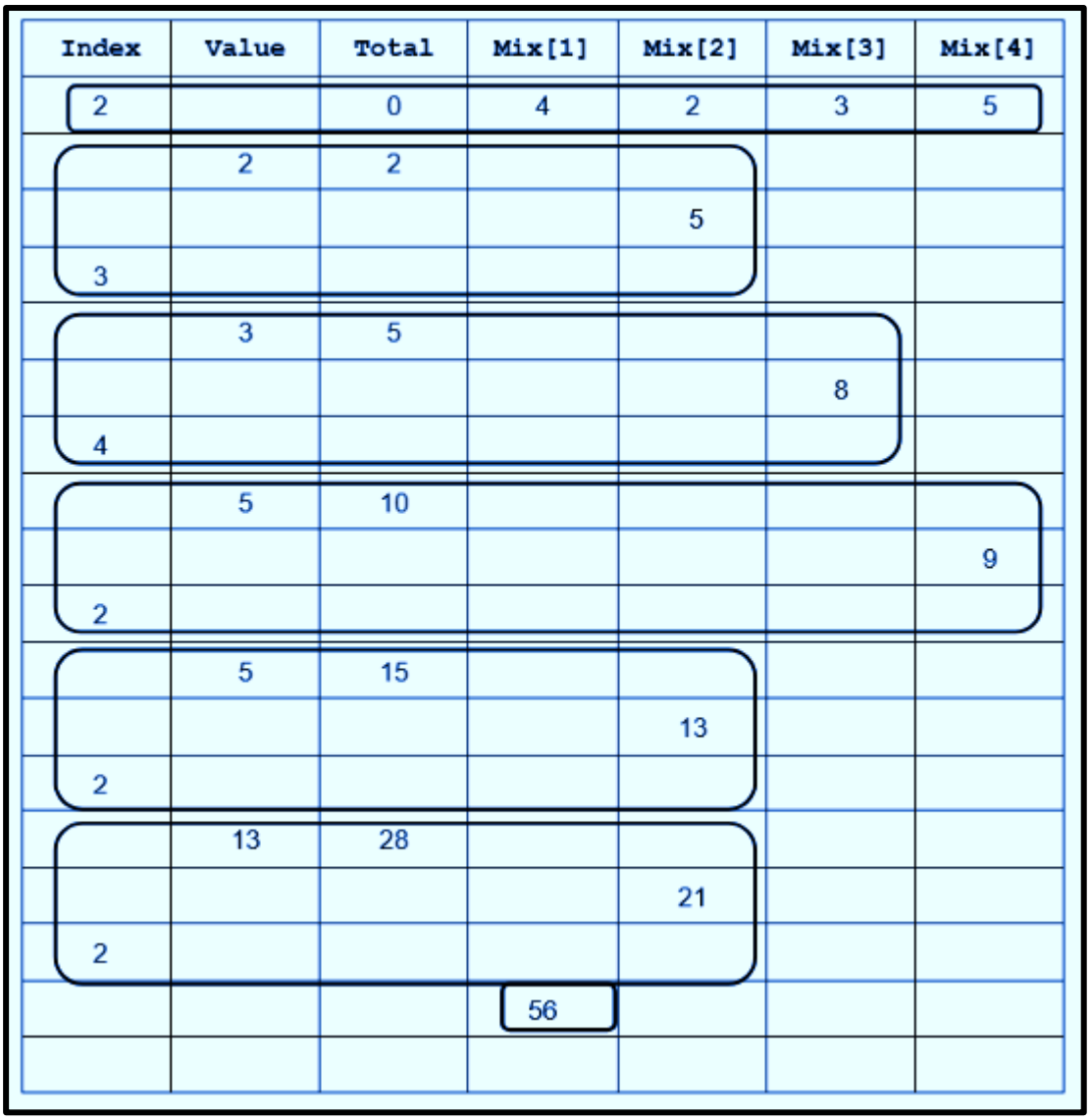
```
PROCEDURE Process(Start : INTEGER)
   DECLARE Value, Index, Total : INTEGER

   Index ← Start
   Total ← 0

   WHILE Total < 20
      Value ← Mix[Index]
      Total ← Total + Value

      IF Index < 4 THEN
         Mix[Index] ← Mix[Index] + Mix[Index+1]
      ELSE
         Mix[Index] ← Mix[Index] + Mix[1]
      ENDIF
      Index ← (Value MOD 4) + 1

   ENDWHILE

   Mix[1] ← Total * Index

ENDPROCEDURE
```

Complete the trace table on the opposite page by dry running the procedure when it is called as follows:

```
CALL Process(2)
```

Index	Value	Total	Mix[1]	Mix[2]	Mix[3]	Mix[4]

[6]



Index	Value	Total	Mix[1]	Mix[2]	Mix[3]	Mix[4]
2		0	4	2	3	5
	2	2				
				5		
3						
	3	5				
					8	
4						
	5	10				
						9
2						
	5	15				
				13		
2						
	13	28				
				21		
2						
			56			

6 A function `TestNum()` will take a six-digit string as a parameter.

The function will test whether the string meets certain conditions and will return an integer value as follows:

Return value	Condition	Example
1	The last three digits are the same but non-zero.	"253444"
2	The last three digits are zero.	"253000"
3	The first three and last three digits are the same.	"410410"

The function will return the highest possible value for the given string.

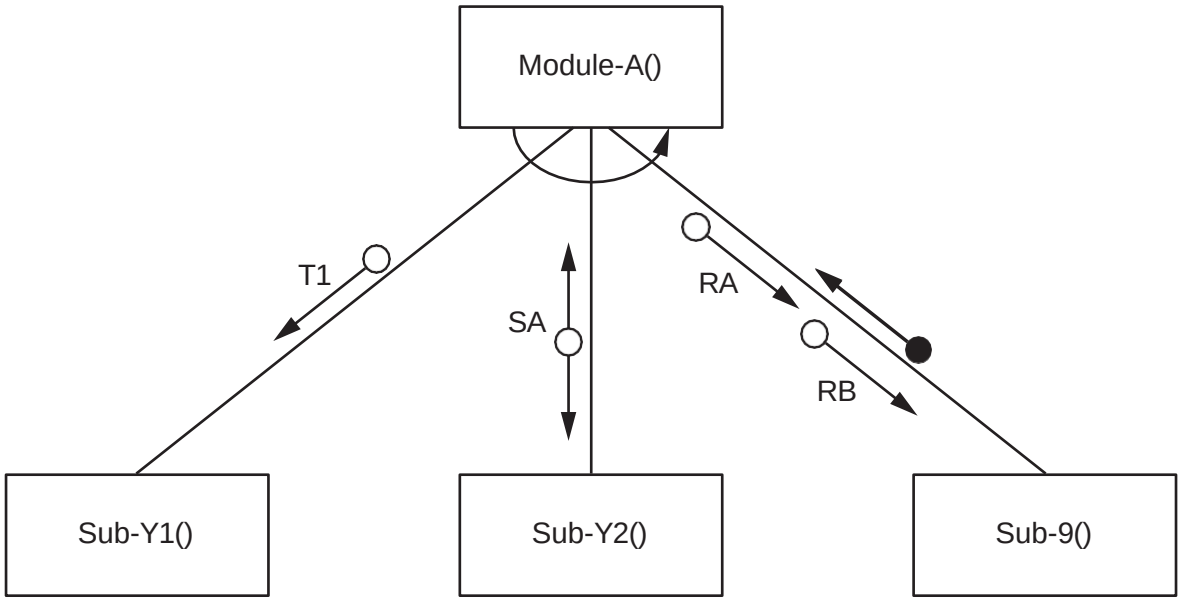
If the string does not meet any of the conditions, zero is returned.

Write pseudocode for function TestNum(). Assume that the parameter is valid.

```
FUNCTION TestNum(ThisNum : STRING) RETURNS INTEGER

   IF LEFT(ThisNum,3) = RIGHT(ThisNum 3) THEN
      RETURN 3
   ENDIF

   IF RIGHT(ThisNum, 3) = "000" THEN
      RETURN 2
   ENDIF

   IF MID(ThisNum, 4, 1) = MID(ThisNum, 5, 1)__
      AND MID(ThisNum, 5, 1) = MID(ThisNum, 6, 1) THEN
      RETURN 1
   ENDIF

   RETURN 0

ENDFUNCTION
```

[6]

7 A structure chart shows the modular structure of a program:



(a) Explain the meaning of the curved arrow symbol which begins and ends at Module-A().

MP1 iteration / looping
MP2 naming all four modules correctly in the correct sequence // e.g. `Module-A` repeatedly calls `Sub-Y1`, then `SubY2` then `Sub-9`

[2]

(b) The structure chart shows that Sub-9() is a function.

A Boolean value is returned by Sub-9() for processing by Module-A().

The original parameter RA is of type integer and RB is of type string.

A record type `MyType` will be defined with three fields to store the values passed between the two modules.

(i) Write pseudocode to define `MyType`.

```
TYPE MyType
   DECLARE RA   : INTEGER
   DECLARE RB   : STRING
   DECLARE RC   : BOOLEAN
ENDTYPE
```

[3]

(ii) The design is modified and Sub-9() is changed to a procedure.

The procedure will be called with a single parameter of type `MyType`.

Write the pseudocode header for procedure `Sub-9()`.

MP 1 One mark for `BYREF`
MP 2 One mark for the rest of the statement

[2]

8 A class of students are developing a program to send data between computers. Many computers are connected together to form a wired network. Serial ports are used to connect one computer to another.

Each computer:
- is assigned a unique three-digit ID
- has three ports, each identified by an integer value
- is connected to between one and three other computers.

Messages are sent between computers as a string of characters organised into fields as shown:

```
<STX><DestinationID><SourceID><Data><ETX>
```

Field name	Description
`STX`	a single character marking the start of the message (ASCII value 02)
`DestinationID`	three numeric characters identifying the destination computer
`SourceID`	three numeric characters identifying the source computer
`Data`	a variable length string containing the data being sent (Minimum length is 1 character)
`ETX`	a single character marking the end of the message (ASCII value 03)

For example, the following message contains the data "Hello Jack" being sent from computer "202" to computer "454":

```
<STX>"454202Hello Jack"<ETX>
```

Each computer will run a copy of the same program. Each program will contain a global variable `MyID` of type string which contains the unique ID of the computer in which the program is running.

The first two program modules are defined as follows:

Module	Description
`GetData()` (already written)	• returns the data field from a message that has been received • If no message is available, the module waits until one has been received.
`ReceiveFile()`	• takes a file name as a parameter of type string • creates a text file with the given file name (no checking required) • writes the data field returned by `GetData()` to the file • repeats until the data field is "****", which is not written to the file • outputs a final message giving the total number of characters written to the file, for example: `132456 characters were written to newfile.txt`

(a) Write pseudocode for module `ReceiveFile()`.

Module `GetData()` has already been written and must be used.

```
PROCEDURE ReceiveFile(FileName : STRING)
   DECLARE FileData     : STRING
   DECLARE CharCount    : INTEGER
   CONSTANT Terminator = "****"

   OPENFILE FileName FOR WRITE
   FileData ← GetData()
   CharCount ← 0

   WHILE FileData <> Terminator
      WRITEFILE FileName, FileData
      CharCount ← CharCount + LENGTH(FileData)
      FileData ← GetData()
   ENDWHILE

   CLOSEFILE FileName

   OUTPUT CharCount (, " characters were written to ",
                                                FileName)

ENDPROCEDURE
```

[7]

(b) The use of the string "****" as explained in the module description for `ReceiveFile()` may cause a problem.

Explain the problem and suggest a solution.

Problem
- If the file being sent contains a line of the string "****"
- then the file being written by `ReceiveFile()` will end at this point // subsequent file lines will be lost

Solution
- Read the file (at the sending end) to find the number of lines it contains
- Send an initial message which defines the number of lines in the file

[3]

(c) Two new modules are defined, which will allow two users to exchange messages.

Module	Description
`Transmit()` (already written)	• takes two parameters: ◦ a string representing a message ◦ an integer representing a port number • transmits the message using the given port
`Chat()`	• takes two parameters: ◦ a string representing a Destination ID ◦ an integer representing a port number • extracts data from a received message using `GetData()` and outputs it • forms a message using data input by the user and sends it using `Transmit()` • repeats until either the output string or the sent string is "`Bye`"

Reminders:

- Each program contains a global variable `MyID` of type string which contains the unique ID of the computer in which the program is running.
- Messages are sent between computers as a string of characters organised into fields as shown:

```
<STX><DestinationID><SourceID><Data><ETX>
```

Write pseudocode for module `Chat()`.

Modules `GetData()` and `Transmit()` must be used.

```
PROCEDURE Chat(Destination : STRING, Port : INTEGER)
   DECLARE Data      : STRING
   DECLARE Finished  : BOOLEAN
   CONSTANT Terminator = "Bye"
   CONSTANT STX = CHR(2)
   CONSTANT ETX = CHR(3)

   Finished ← FALSE

   REPEAT
      Data ← GetData()
      OUTPUT Data
      IF Data = Terminator THEN
         Finished ← TRUE
      ENDIF

      IF NOT Finished THEN    //about to reply
         INPUT Data
         Transmit(STX & Destination & MyID & Data & ETX,
                                                  Port)
         IF Data = Terminator THEN
            Finished ← TRUE
         ENDIF
      ENDIF

   UNTIL Finished = TRUE

ENDPROCEDURE
```

[7]

(d) Module `GetData()` returns the data field from a message that has been received. If no message is available, the module waits until one has been received.

Explain the limitation of this on module `Chat()` from part **(c)**.

Describe a modification to `GetData()` to address this limitation.

Limitation .....

1 `GetData()` does not return a value until a message has been received
2 So once a message has been sent the user has to wait for a reply // chat is half-duplex

.....

Modification .....

3 If no response allow the receiver to exit chat at any time …
4 `GetData()` should immediately return a suitable message // set a time limit
5 … which `Chat()` can detect and respond by allowing the conversation to continue

[3]