Cambridge Assessment International Education

Cambridge International AS & A Level

CANDIDATE NAME

CENTRE NUMBER

CANDIDATE NUMBER

3141164855

COMPUTER SCIENCE **9618/23**

Paper 2 Fundamental Problem-solving and Programming Skills **May/June 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. Any blank pages are indicated.

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

1 The following pseudocode represents part of the algorithm for a program.

Line numbers are for reference only.

```
10   DECLARE Sheet4 : ARRAY[1:2, 1:50] OF INTEGER

…

100  FOR PCount ← 0 TO 49
101      Sheet4[1, PCount] ← 0
102      Sheet4[2, PCount] ← 47
103  NEXT PCount
```

(a) The pseudocode contains references to an array.

Complete the table by writing the answer for each row.

	Answer
The dimension of the array	2
The name of the variable used as an array index	PCount
The number of elements in the array	100

[3]

(b) The pseudocode contains two errors. One error is that variable `PCount` has not been declared.

Identify the **other** error **and** state the line number where it occurs.

Error .... The (second dimension/index of the) array is declared from 1 to 50 but the loop runs from 0 to 49 ....

....

....

Line number . Line number: 10 / 100 / 101 / 102 .... [2]

(c) The pseudocode does not include a declaration for `PCount`.

State the data type that should be used in the declaration.

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

(d) The pseudocode statements given in the following table are used in other parts of the algorithm.

Complete the table by placing **one or more** ticks (✓) in each row.

The first row has already been completed.

Pseudocode statement	**Input**	**Process**	**Output**
`INPUT MyChoice`	✓		
`OUTPUT FirstName & LastName`		✓	✓
`WRITEFILE YourFile, TextLine`			✓
`READFILE MyFile, TextLine`	✓		
`Result ← SQRT(NextNum)`		✓	

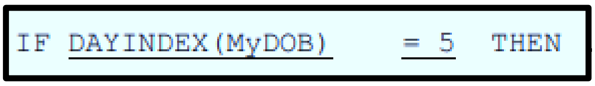
[4]

2 A program stores a date of birth for a student using a variable, `MyDOB`, of type `DATE`.

(a) `MyDOB` has been assigned a valid value corresponding to Kevin's date of birth.

Complete the pseudocode statement to test whether Kevin was born on a Thursday.

`IF` .... `THEN` [2]



```
IF DAYINDEX(MyDOB)   = 5  THEN
```

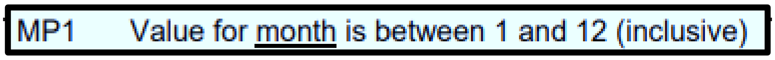
(b) A function `CheckDate()` will take three integer parameters representing a day, month and year of a given date.

The function will validate the date of birth for a student that the parameters passed to it represent.
For a date to be valid, a student must be at least 18 in year 2020.

(i) Two of the parameter values can be checked without reference to the third parameter.

Describe these **two** checks.

Check 1 ....

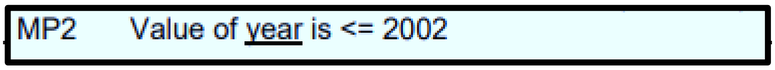


MP1 Value for month is between 1 and 12 (inclusive)

....

....

Check 2 ....



MP2 Value of year is <= 2002

....

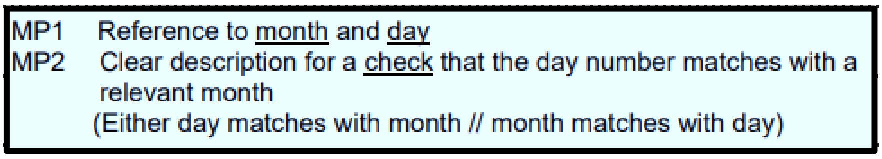
....

.... [2]

(ii) Several values of the parameter representing the day can only be checked completely by referring to the value of **one other** parameter.

Describe this check.

....



MP1 Reference to month and day
MP2 Clear description for a check that the day number matches with a relevant month
(Either day matches with month // month matches with day)

... .

....

.... [2]

BLANK PAGE

3 A program processes data using a stack. The data is copied to a text file before the program ends.

(a) The following diagram shows the current state of the stack.

The operation of this stack may be summarised as follows:

- The `TopOfStack` pointer points to the last item added to the stack.
- The `BottomOfStack` pointer points to the first item on the stack.
- The stack grows upwards when items are added.

Stack **Pointer**

Memory location	Value	
506		
505	WWW	← `TopOfStack`
504	YYY	
503	XXX	
502	ZZZ	
501	NNN	
500	PPP	← `BottomOfStack`

(i) An error will be generated if an attempt is made to POP a value when the stack is empty.

State the maximum number of consecutive POP operations that could be performed on the stack shown above **before** an error is generated.

.... 6 .... [1]

(ii) The following operations are performed:

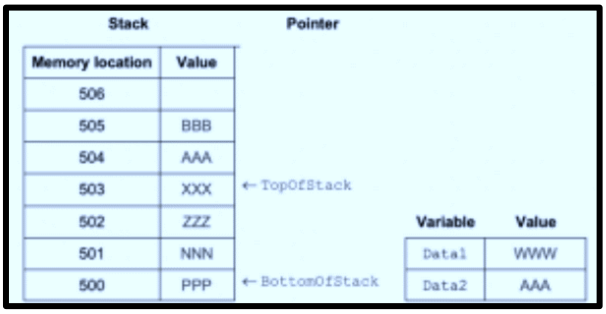
1. POP and store value in variable `Data1`
2. POP and store value in variable `Data2`
3. PUSH value AAA
4. PUSH value BBB
5. POP and discard value
6. POP and store value in variable `Data2`

Complete the diagram to show the state of the stack and the variables **after** the given operations have been performed.

Stack

Memory location	Value
506	
505	
504	
503	
502	
501	
500	

Pointer



Stack Memory location	Value	Pointer
506		
505	BBB	
504	AAA	
503	XXX	← TopOfStack
502	ZZZ	
501	NNN	
500	PPP	← BottomOfStack

Variable	Value
Data1	WWW
Data2	AAA

Variable	Value
Data1	
Data2	

[4]

(b) The data is copied to a text file before the program ends.

(i) State an advantage of writing the data from the stack to a text file before the program ends.

So that the data may be recovered / restored (the next time the program is run)
// the data is permanently saved / data is not lost when the program terminates

..... [1]

(ii) A module `SaveStack()` will write the data from the stack to a text file.

Express an algorithm for `SaveStack()` as five steps that could be used to produce pseudocode.

Write the **five** steps.

Step 1 .....

MP1 Open the text file in WRITE mode
MP2 Check there is a value on the stack
MP3 POP value ....
MP4 Write value to the text file
MP5 Repeat from Step 2 // loop referencing the stack items

Alternative solution: Not using POP primitive

MP1 Open the text file in WRITE mode
MP2 Check there is a value on the stack
MP3 Read value from ToS location
MP4 Write the value to the text file – Must some attempt at 'the value' NOT '<u>all</u> the values'
MP5 Decrement ToS
MP6 Repeat from step 2 // loop referencing the stack items

Step 2 .....

Step 3 .....

Step 4 .....

Step 5 .....

..... [5]

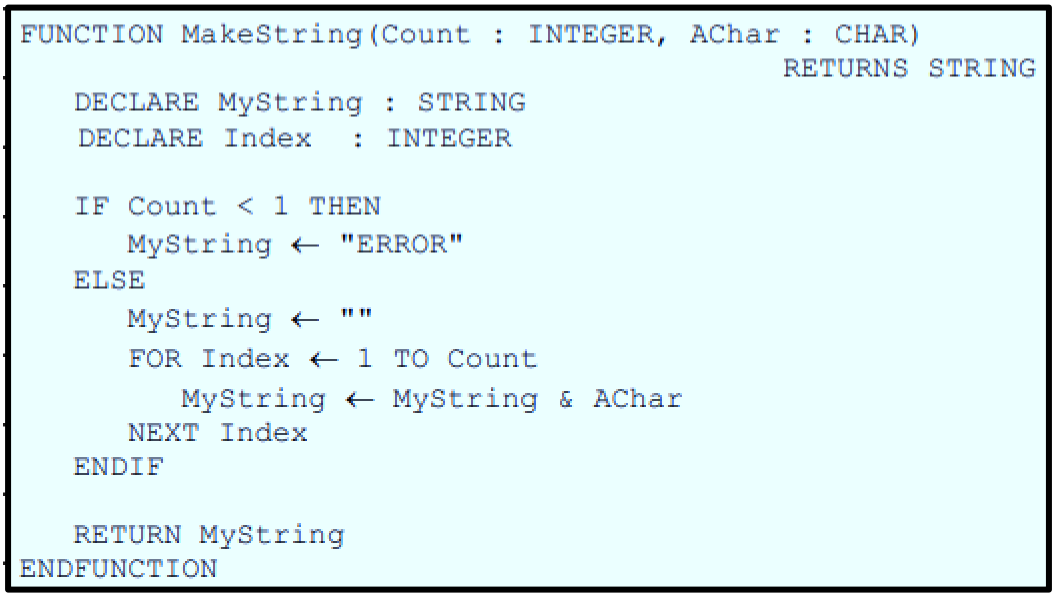
4 A function `MakeString()` will:

1. take two parameters:
   - a count as an integer
   - a character
2. generate a string of length equal to the count, made up of the character
3. return the string generated, or return `"ERROR"` if the count is less than 1.

For example, the function call:

`MakeString(3, 'Z')` will return the string `"ZZZ"`

Write pseudocode for function `MakeString()`.



```
FUNCTION MakeString(Count : INTEGER, AChar : CHAR)
                                          RETURNS STRING
   DECLARE MyString : STRING
   DECLARE Index    : INTEGER

   IF Count < 1 THEN
      MyString ← "ERROR"
   ELSE
      MyString ← ""
      FOR Index ← 1 TO Count
         MyString ← MyString & AChar
      NEXT Index
   ENDIF

   RETURN MyString
ENDFUNCTION
```

.... [6]

5 A program is designed, coded and compiled without errors. The compiled code is sent for testing.

(a) The program will be tested using the walkthrough method.

Additional information will be needed before this method can be used.

Identify this additional information **and** explain why it is needed.

Additional information

Additional Information:
MP1 The (program/source) code/specification
MP2 test plan // inputs/test data and expected outputs

Explanation

Explanation:
MP3 The structure / design / algorithm of the program of the program needs to be known
MP4 …. so that all paths through the program can be tested

.... [3]

(b) Testing is completed and the program is made available to users.

Some time later, changes are made to the program to improve the speed of response.

State the type of maintenance that has been applied to the program.

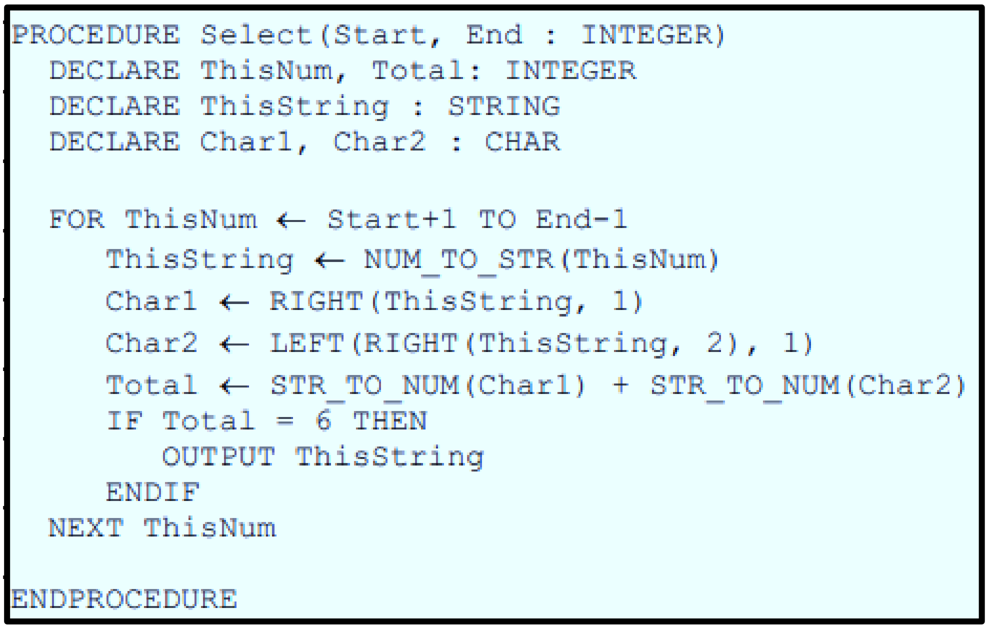
..........Perfective.. [1]

6 A procedure `Select()` will:

1. take two integer values as parameters representing start and end values where both values are greater than 9 and the end value is greater than the start value
2. output **each** integer value between the start and the end value (**not** including the start and end values), where the sum of the last two digits is 6, for example, 142.

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

Parameter validation is **not** required.



```
PROCEDURE Select(Start, End : INTEGER)
  DECLARE ThisNum, Total: INTEGER
  DECLARE ThisString : STRING
  DECLARE Char1, Char2 : CHAR

  FOR ThisNum ← Start+1 TO End-1
     ThisString ← NUM_TO_STR(ThisNum)
     Char1 ← RIGHT(ThisString, 1)
     Char2 ← LEFT(RIGHT(ThisString, 2), 1)
     Total ← STR_TO_NUM(Char1) + STR_TO_NUM(Char2)
     IF Total = 6 THEN
        OUTPUT ThisString
     ENDIF
  NEXT ThisNum

ENDPROCEDURE
```

[7]

(b) The check performed by procedure `Select()` on the last two digits is needed at several places in the program and will be implemented using a new function.

The new function `CheckNum()` will:

- allow the required sum to be specified (not just 6)
- check one number
- return an appropriate value.

Describe the function interface **and two** advantages of this modular approach.

Interface ......

....

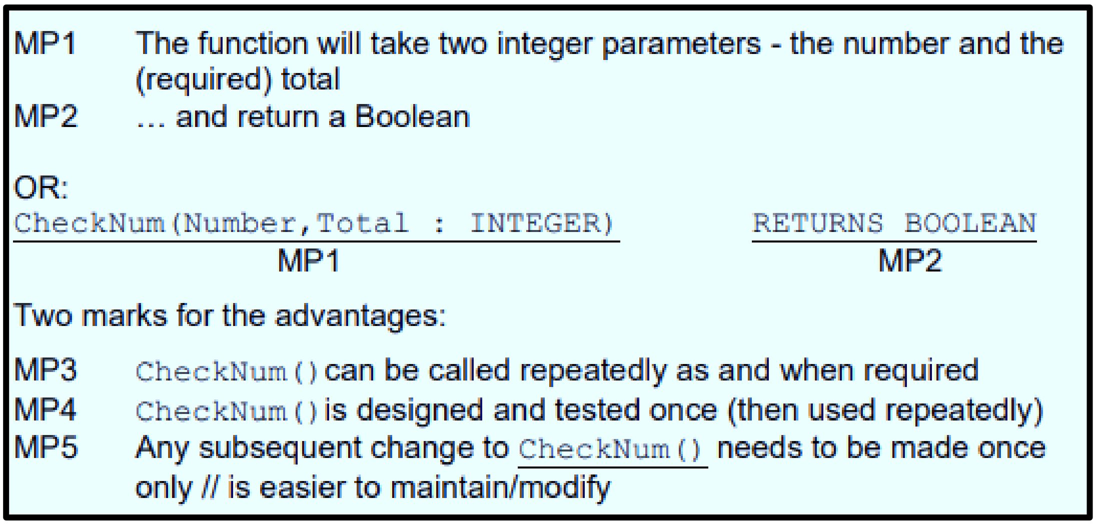
....

....

Advantage 1

....

Advantage 2



MP1 The function will take two integer parameters - the number and the (required) total

MP2 ... and return a Boolean

OR:

`CheckNum(Number,Total : INTEGER)` `RETURNS BOOLEAN`

$\underbrace{\texttt{CheckNum(Number,Total : INTEGER)}}_{\text{MP1}}$ $\underbrace{\texttt{RETURNS BOOLEAN}}_{\text{MP2}}$

Two marks for the advantages:

MP3 `CheckNum()` can be called repeatedly as and when required

MP4 `CheckNum()` is designed and tested once (then used repeatedly)

MP5 Any subsequent change to `CheckNum()` needs to be made once only // is easier to maintain/modify

....

[4]

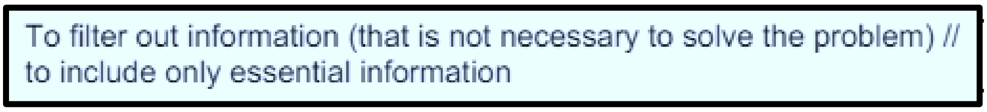
7 A school has a library system which allows students to borrow books for a length of time. Information relating to students and books is stored in text files. Student information includes name, home address, email address, date of birth, tutor and subject choices. Book information includes author, title, subject category, library location and the date that the book was borrowed.

A program helps the staff to manage the borrowing of books.

(a) A new module needs to be written to generate emails to send to students who have an overdue book. Students who are sent an email are prevented from borrowing any more books until the overdue book is returned.

The process of abstraction has been used when designing the new module.

(i) State the purpose of applying abstraction to this problem.



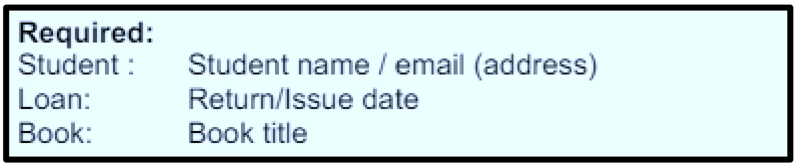
To filter out information (that is not necessary to solve the problem) // to include only essential information

.... [1]

(ii) Identify **one** item of information that is required and **one** item that is **not** required in the new module. Justify your choices.

Item required ....

Justification ....



Required:

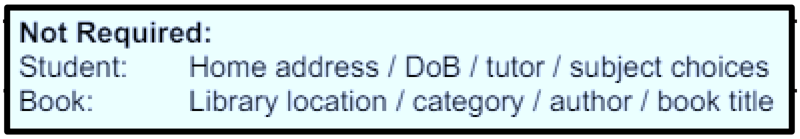
Student :	Student name / email (address)
Loan:	Return/Issue date
Book:	Book title

....

....

Item not required ....

Justification ....



Not Required:

Student:	Home address / DoB / tutor / subject choices
Book:	Library location / category / author / book title

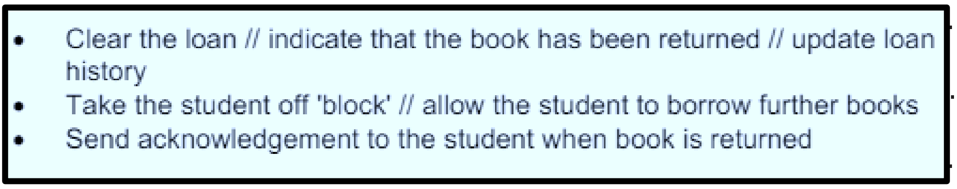
....

.... [2]

(iii) Identify **two** operations that would be required to process data when an overdue book is returned.

Operation 1 ..

Operation 2 ..



- Clear the loan // indicate that the book has been returned // update loan history
- Take the student off 'block' // allow the student to borrow further books
- Send acknowledgement to the student when book is returned

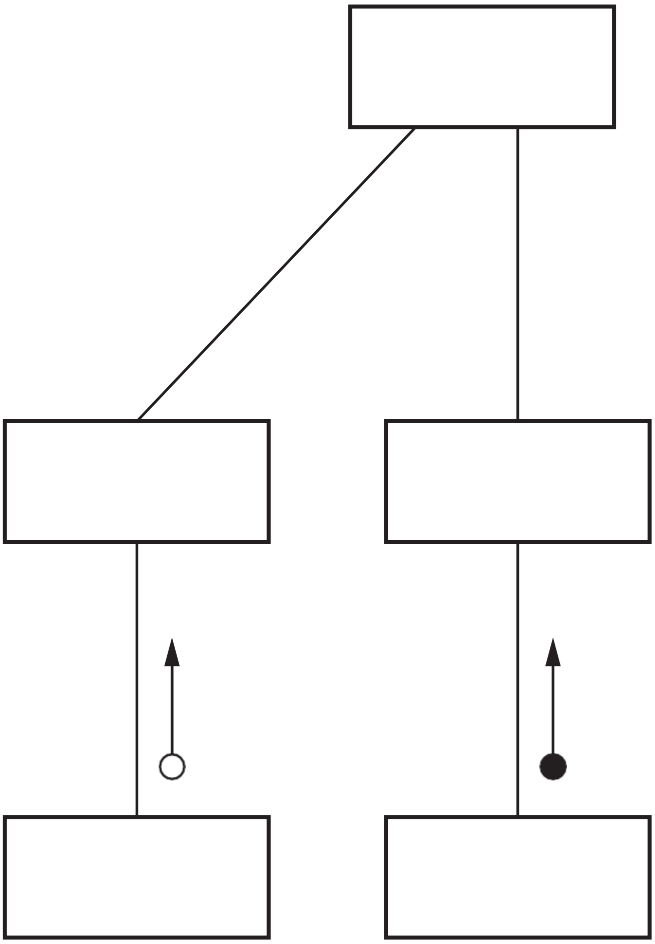
.... [2]

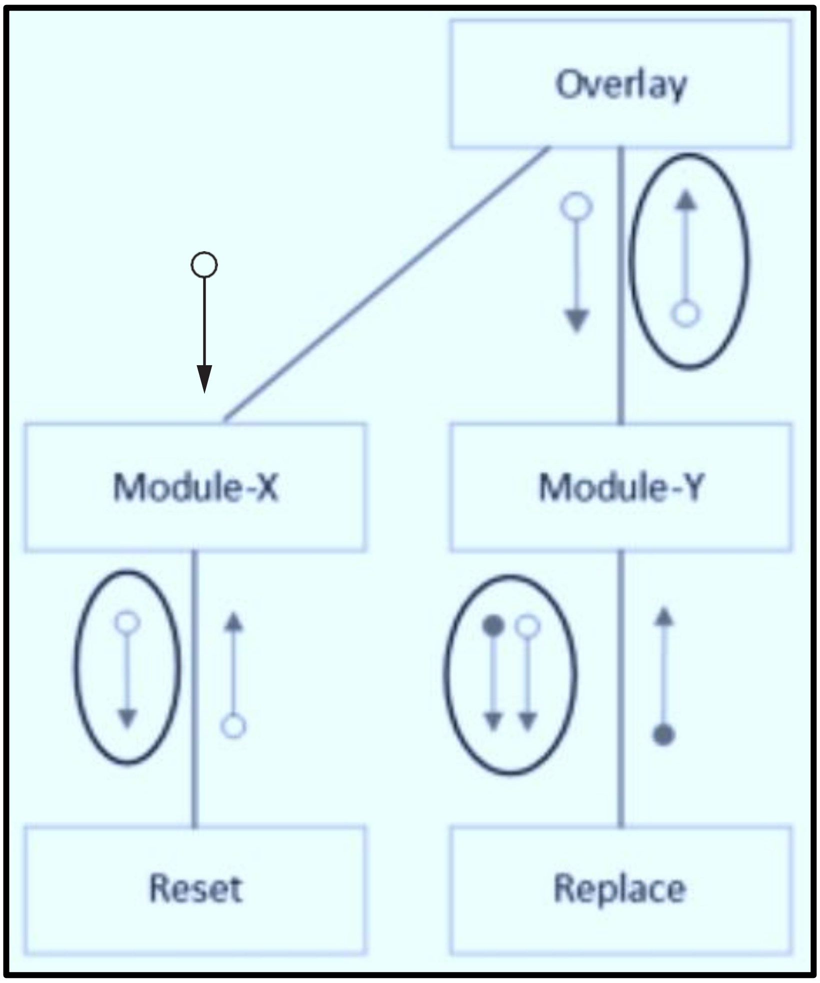
(b) Part of the library program contains program modules with headers as follows:

Pseudocode module header
`PROCEDURE Module-X()`
`PROCEDURE Module-Y(BYREF RA : INTEGER, SA : REAL)`
`PROCEDURE Overlay()`
`FUNCTION Replace(RA : INTEGER, RB : BOOLEAN) RETURNS BOOLEAN`
`FUNCTION Reset(TA : STRING) RETURNS INTEGER`

`Module-X()` and `Module-Y()` are both called from module `Overlay()`.

Complete the structure chart.





8 A computer shop assembles desktop computers, using items bought from several suppliers. A text file `Stock.txt` contains information about each item.

Information for each item is stored as a single line in the `Stock.txt` file in the format:

```
<ItemNum><SupplierCode><Description>
```

Item information is as follows:

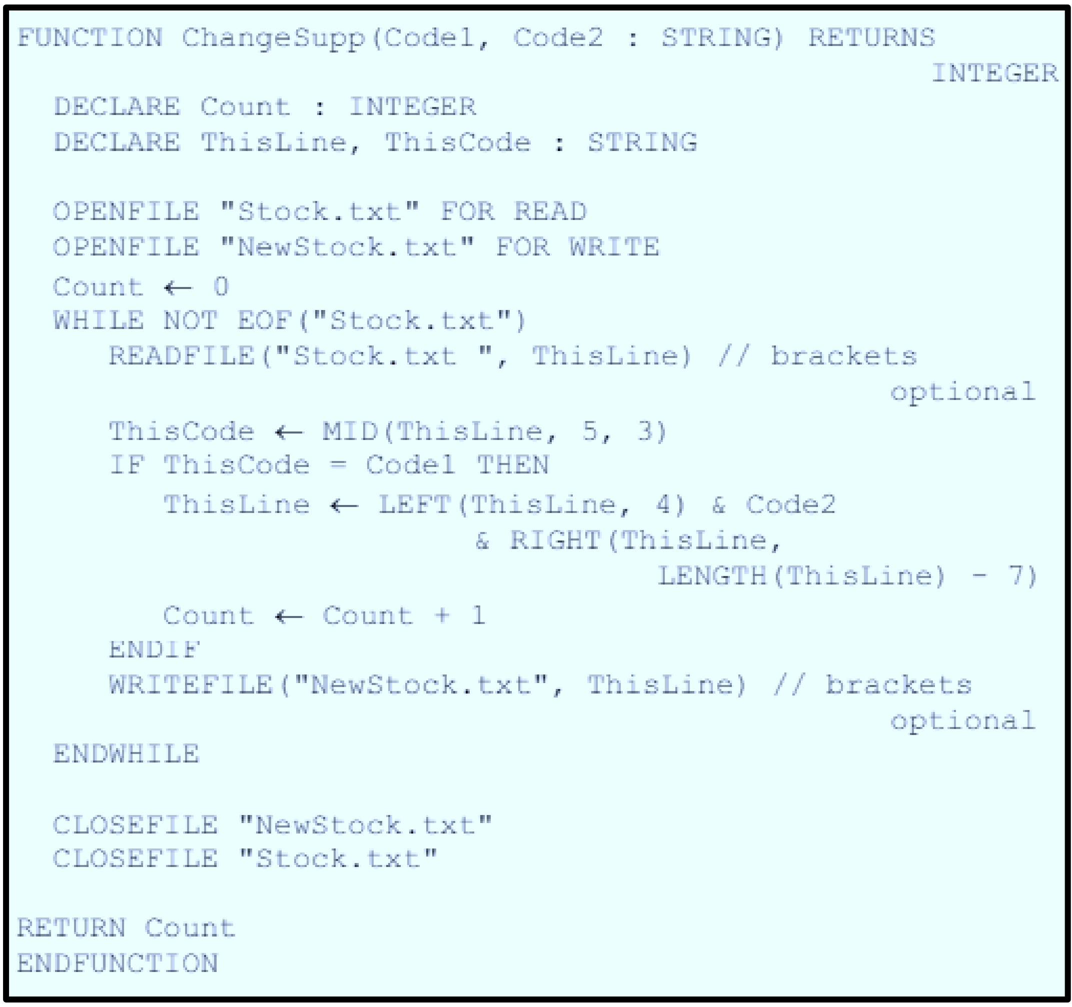
	Format	Comment
`ItemNum`	4 numeric characters	unique number for each item in the range “0001” to “5999” inclusive
`SupplierCode`	3 alphabetic characters	code to identify the supplier of the item
`Description`	a string	a minimum of 12 characters

The file is organised in ascending order of `ItemNum` and does not contain all possible values in the range.

The programmer has defined the first program module as follows:

Module	Description
`ChangeSupp()`	• called with two parameters `Code1` and `Code2` of type string that represent valid supplier codes • creates a new file `NewStock.txt` from the contents of the file `Stock.txt` where any reference to `Code1` is replaced by `Code2` • returns a count of the number of items that have had their supplier code changed

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



```
FUNCTION ChangeSupp(Code1, Code2 : STRING) RETURNS
                                                INTEGER
  DECLARE Count : INTEGER
  DECLARE ThisLine, ThisCode : STRING

  OPENFILE "Stock.txt" FOR READ
  OPENFILE "NewStock.txt" FOR WRITE
  Count ← 0
  WHILE NOT EOF("Stock.txt")
     READFILE("Stock.txt ", ThisLine) // brackets
                                              optional
     ThisCode ← MID(ThisLine, 5, 3)
     IF ThisCode = Code1 THEN
        ThisLine ← LEFT(ThisLine, 4) & Code2
                         & RIGHT(ThisLine,
                                  LENGTH(ThisLine) - 7)
        Count ← Count + 1
     ENDIF
     WRITEFILE("NewStock.txt", ThisLine) // brackets
                                              optional
  ENDWHILE

  CLOSEFILE "NewStock.txt"
  CLOSEFILE "Stock.txt"

RETURN Count
ENDFUNCTION
```

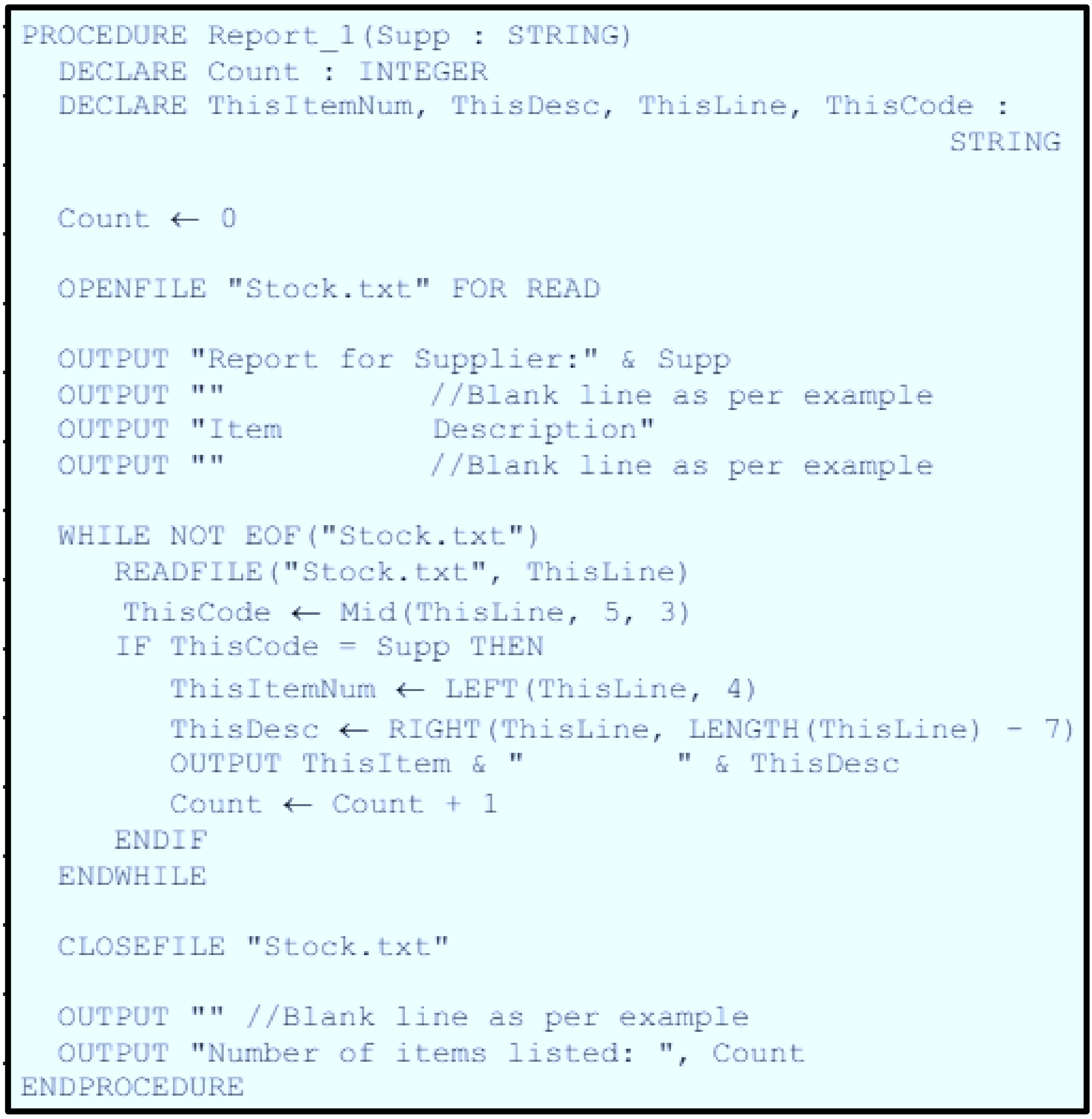
[8]

(b) A new module is required:

Module	Description

`Report_1()`	• takes a parameter of type string that represents a `SupplierCode`
	• searches the `Stock.txt` file for each line of item information that contains the given `SupplierCode`
	• produces a formatted report of items for the given `SupplierCode`, for example, for supplier DRG, the output could be:
	`Report for Supplier: DRG`
	`Item        Description`
	`1234        USB Printer Cable 3m`
	`1273        32GB USB Flash Drive`
	`1350        Mouse Mat 320 x 240mm`
	`Number of items listed: 3`

Write pseudocode for module `Report_1()`.



```
PROCEDURE Report_1(Supp : STRING)
  DECLARE Count : INTEGER
  DECLARE ThisItemNum, ThisDesc, ThisLine, ThisCode :
                                                    STRING

  Count ← 0

  OPENFILE "Stock.txt" FOR READ

  OUTPUT "Report for Supplier:" & Supp
  OUTPUT ""             //Blank line as per example
  OUTPUT "Item          Description"
  OUTPUT ""             //Blank line as per example

  WHILE NOT EOF("Stock.txt")
     READFILE("Stock.txt", ThisLine)
     ThisCode ← Mid(ThisLine, 5, 3)
     IF ThisCode = Supp THEN
        ThisItemNum ← LEFT(ThisLine, 4)
        ThisDesc ← RIGHT(ThisLine, LENGTH(ThisLine) - 7)
        OUTPUT ThisItem & "         " & ThisDesc
        Count ← Count + 1
     ENDIF
  ENDWHILE

  CLOSEFILE "Stock.txt"

  OUTPUT "" //Blank line as per example
  OUTPUT "Number of items listed: ", Count
ENDPROCEDURE
```

....

....

....

....

....

....

....

....

....

....

....

....

....

....

.... [6]

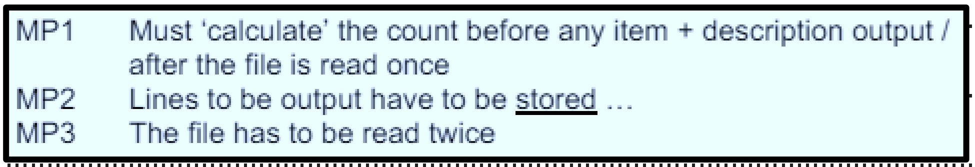
(c) The format of the output from module `Report_1()` from **part (b)** is changed. The number of items listed is moved to the top of the report as shown in the example:

```
Report for Supplier: DRG
Number of items listed: 3

Item        Description

1234        USB Printer Cable 3m
1273        32GB USB Flash Drive
1350        Mouse Mat 320 x 240mm
```

(i) Explain why this new layout would increase the complexity of the algorithm.

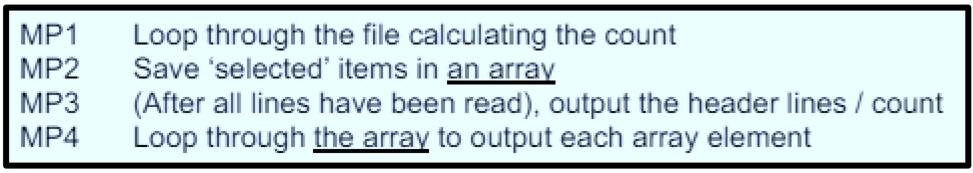


MP1 Must 'calculate' the count before any item + description output / after the file is read once
MP2 Lines to be output have to be stored …
MP3 The file has to be read twice

.... [2]

(ii) The algorithm will be modified to produce the report in the new format. The modified algorithm will be implemented so that the file `Stock.txt` is only read once.

Describe the modified algorithm.



MP1 Loop through the file calculating the count
MP2 Save 'selected' items in an array
MP3 (After all lines have been read), output the header lines / count
MP4 Loop through the array to output each array element

....

....

....

.... [3]

BLANK PAGE

BLANK PAGE