AS · Cambridge (CIE) · Computer Science

3 hours 27 questions

Exam Questions

Structured Programming

Procedures & functions

1 A program uses three global integer variables HH, MM and SS to represent the current time in hours, minutes and seconds using the 24-hour clock notation.

Midnight would be represented as 00:00:00 (HH:MM:SS). If the variables HH, MM and SS contained the values 16, 30 and 10 respectively, then the time would be 16:30:10 or just after 4.30 in the afternoon.

A procedure Tick() will be called every second.
The procedure Tick() will:

- update the value in SS each time it is called
- update the values in HH and MM as appropriate
- call a procedure CheckAlarm() at the start of each minute
- call a procedure NewDay() whenever the time reaches midnight.

Complete the pseudocode for procedure Tick().

```
PROCEDURE Tick()
```

Answer

Mark Scheme and Guidance

Example solution:

```
PROCEDURE Tick()
SS ← SS + 1
IF SS = 60 THEN
SS ← 0
MM ← MM + 1
IF MM = 60 THEN
MM ← 0
HH ← HH + 1
IF HH = 24 THEN
HH ← 0
CALL NewDay()
ENDIF
```

```
ENDIF
CALL CheckAlarm()
ENDIF

ENDPROCEDURE
```

Mark as follows:

1. Increment SS anywhere in the code
2. Call CheckAlarm() when SS is set to zero
3. Test if SS = 60 anywhere in the code ... if so:
4. set SS to 0
5. increment MM
6. When MM is set to 60 set MM to 0 **and** increment HH
7. When HH is set to 24 set HH to 0 **and** call NewDay()

Max 6 marks

(6 marks)

2 (a) A program is being developed to process bank card information.

When a card number is displayed, all the characters except the last four are replaced with the asterisk character '*'.

Card numbers are stored as strings. The strings are between 10 and 20 characters in length.

The function Conceal() will take a string representing a card number and return a modified string.

Example strings:

Original string	**Modified string**
"1234567890"	"******7890"
"1234567897652"	"*********7652"
"1234567890123456"	"************3456"

The function Conceal() will:

- take a numeric string as a parameter representing the card number
- return a string in which the asterisk character replaces all except the last four characters of the card number parameter.

Write pseudocode for the function Conceal().

Answer

Mark Scheme and Guidance

Example 'loop solution':

```
FUNCTION Conceal(CardNumber : STRING) RETURNS STRING
DECLARE MaskedString : STRING
DECLARE Count : INTEGER
CONSTANT Asterisk = '*'

MaskedString ← RIGHT(CardNumber, 4)
FOR Count ← 1 TO LENGTH(CardNumber) - 4
MaskedString ← Asterisk & MaskedString
NEXT Count

RETURN MaskedString

ENDFUNCTION
```

Mark as follows:
MP1 Function heading and parameter and ending and return type
MP2 Declaration of all local variables used - including the loop counter
MP3 Calculate number of digits to mask / number of asterisks required
MP4 use of a 'relevant' Loop
MP5 Correct number of iterations
MP6 Concatenate asterisk to start/end of MaskedString in a loop
MP7 Assign last four characters to MaskedString // Concatenate the retained original last four digits
MP8 Return masked string

Max 6 marks

ALTERNATIVE ‘non loop’ solution:

```
FUNCTION Conceal(CardNumber : STRING) RETURNS STRING
DECLARE MaskedString : STRING
DECLARE Count : INTEGER
CONSTANT Asterisks = "********************" //20 asterisks
Count  ← LENGTH(CardNumber) - 4
MaskedString  ←  LEFT(Asterisks, Count) & RIGHT(CardNumber, 4)
RETURN MaskedString

ENDFUNCTION
```

Mark as follows:
MP1 Function heading **and** parameter **and** ending **and** return type
MP2 Declaration of all local variables used
MP3 Calculate number of digits to mask / number of asterisks required
MP4 Trim asterisk string to number calculated in **MP3**
MP5 Extract the last four characters
MP6 Concatenate trimmed asterisk string with last four characters of CardNumber
MP7 Return masked string

Max 6 marks

(6 marks)

(b) The requirements have been changed. Conceal() will now be written as a procedure which will process 100 card numbers each time it is called.

The card numbers will be stored in a 2D array CardNumber. The original string will be stored in column one and the modified string in column two.

The new procedure Conceal() will write the modified string to the corresponding element in column two.

The array CardNumber is passed as a parameter to the new procedure Conceal().

Identify how this parameter should be specified in the new procedure head

Answer

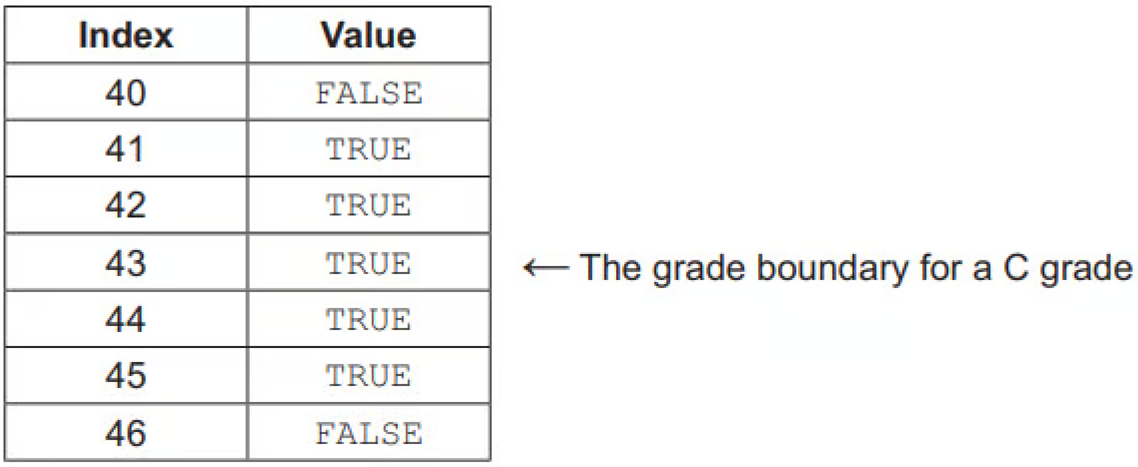
Mark Scheme and Guidance

Any reference to BYREF // 'by reference'

(1 mark)

3 An alternative algorithm to determine if a paper needs to be checked uses a global 1D array Check, containing 76 elements of type BOOLEAN. The indices of the array are from 0 to 75 (inclusive), corresponding to the range of possible marks.

An element value in Check is TRUE if the index is within 2 marks of a grade boundary. For example, in the case where the C grade boundary is 43 the corresponding part of the Check array would be as follows:



Index	Value
40	FALSE
41	TRUE
42	TRUE
43	TRUE
44	TRUE
45	TRUE
46	FALSE

← The grade boundary for a C grade

A procedure GBInitialise() will initialise the Check array using values from the GB array.

Note it can be assumed that the maximum grade boundary value for A is 70 and the minimum value for E is 15.

Write pseudocode for the procedure.

Answer

Mark Scheme and Guidance

Example:solution

```
PROCEDURE GBInitialise()
DECLARE GBIndex, GBMark, ThisIndex : INTEGER

FOR ThisIndex ← 0 TO 75
Check[ThisIndex] ← FALSE // initialise all values
NEXT ThisIndex

FOR GBIndex ← 1 TO 5
GBMark ← GB[GBIndex]
FOR ThisIndex ← GBMark - 2 TO GBMark + 2
```

```
Check[ThisIndex] ← TRUE //Set GBMark ← 2 to TRUE
NEXT ThisIndex
NEXT GBIndex
ENDPROCEDURE
```

Mark as follows:
MP1 Procedure heading and ending
MP2 Initialise Check array to FALSE
MP3 Loop through GB array
MP4 Extract the GB mark
MP5 Loop for 5 elements across GBMark ← 2 // Check if within GBMark ← 2
MP6 Assign each element of Check array to TRUE

ALTERNATIVE – Example using selection Version 1

```
FOR ThisIndex ← 0 TO 75
CASE ThisIndex
GB[1] – 2 TO GB[1] + 2 : Check[ThisIndex] ← TRUE
GB[2] – 2 TO GB[2] + 2 : Check[ThisIndex] ← TRUE
GB[3] – 2 TO GB[3] + 2 : Check[ThisIndex] ← TRUE
GB[4] – 2 TO GB[4] + 2 : Check[ThisIndex] ← TRUE
GB[5] – 2 TO GB[5] + 2 : Check[ThisIndex] ← TRUE
OTHERWISE: Check[ThisIndex] ← FALSE
ENDCASE
NEXT ThisIndex
```

ALTERNATIVE – Example using selection Version 2

```
DECLARE ThisIndex, GBIndex : INTEGER
DECLARE Lower, Higher : INTEGER

FOR ThisIndex ← 0 TO 75
For GBIndex ← 1 T0 5
Lower ← GB[GBIndex] - 2
Higher ← GB[GBIndex] + 2
IF ThisIndex >= Lower AND ThisIndex <= Higher THEN
Check[ThisIndex] ← TRUE
ELSE
Check[ThisIndex] ← FALSE
ENDIF
NEXT Index
NEXT ThisIndex
```

Mark as follows:
MP1 Procedure heading and ending
MP2 Loop through Check array// loop from 0 to 75
MP3 Attempt at using CASE / Selection structure with five clauses/ **five** checks for range
MP4 Extract GB grade
MP5 Compare ThisIndex with each element in GB array ±2
MP6 Assign each element of Check array to TRUE if in range **or** FALSE if not in range

(6 marks)

4 The variable names A, B, C and D in part **(a)** are **not** good programming practice.

(i) State why these variable names are **not** suitable.

[1]

(ii) Identify **one** problem that these variable names might cause.

[1]

(iii) The choice of suitable variable names is one example of good programming practice.

Give one other example.

[1]

Answer

Mark Scheme and Guidance

(i)

The names do not reflect / indicate the purpose of the variable // the names are not meaningful

(ii)

They make the program more difficult to understand / debug / maintain

(iii)

One mark from:

- Indentation / use of white space
- Capitalisation of keywords
- Use of comments
- Use of modular programming
- Use of local variables

Note: max 1 mark

(3 marks)

5 A global 1D array Data contains 100 elements of type integer.

A function Check() will:

- total the element values in odd index locations (1, 3, 5 ... 97, 99)
- total the element values in even index locations (2, 4, 6 ... 98, 100)
- return one of three strings ‘Odd’, ‘Even’ or ‘Same’ to indicate which total is the greater, or whether the totals are the same.

Write pseudocode for the function Check().

Answer

Mark Scheme and Guidance

Example:

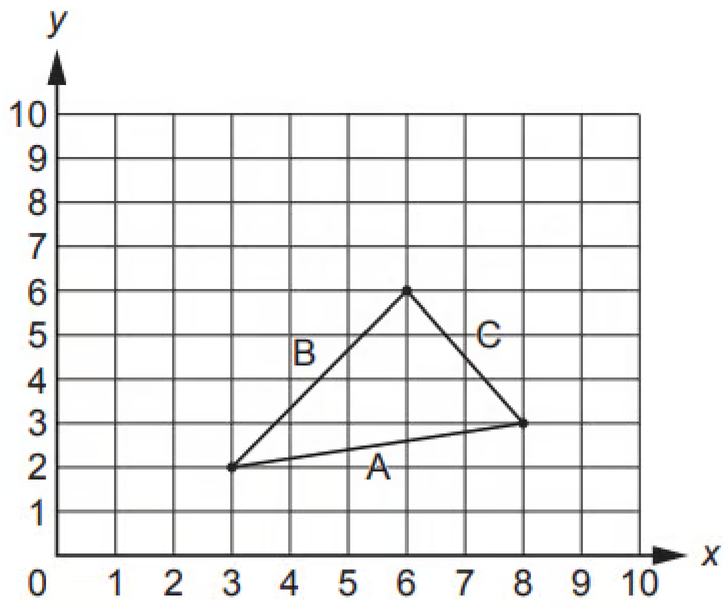
```
FUNCTION Check() RETURNS STRING
DECLARE Odd, Even, Index : INTEGER

Odd ← 0
Even ← 0

FOR Index ← 1 TO 100
IF Index MOD 2 = 0 THEN
Even ← Even + Data[Index]
ELSE
Odd ← Odd + Data[Index]
ENDIF
NEXT Index

ENDFUNCTION
```

Mark as follows:

1. Function heading, ending and return type
2. Declare local variables Odd, Even and Index as integers
3. Initialise Odd and Even
4. Loop for 100 iterations // through array
5. Sum Odd and Even element values **in a loop**
6. Compare Odd and Even after the loop and Return appropriate string

(6 marks)

6 Three points on a grid form a triangle with sides of length A, B and C as shown in the example:



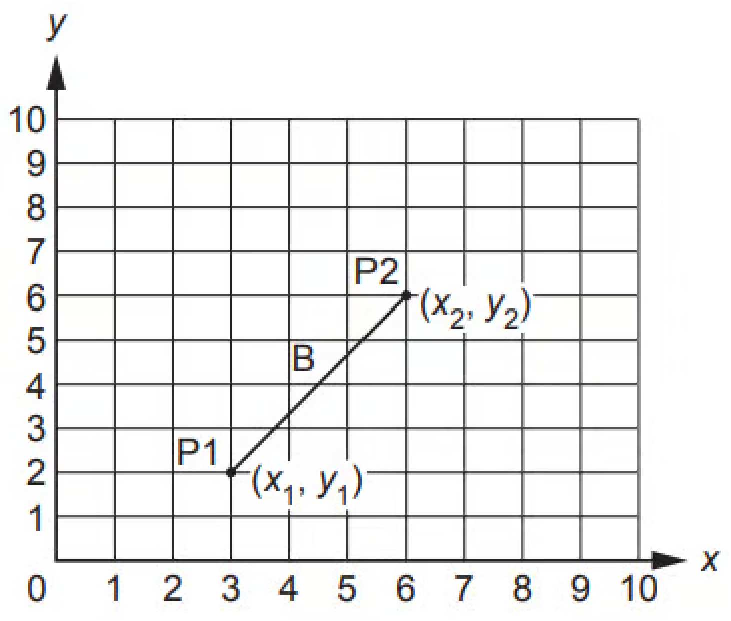
A triangle is said to be right-angled if the following test is true (where A is the length of the longest side):

$A^2 = B^2 + C^2$

A^2 means A multiplied by A, for example 3^2 means 3 × 3 which evaluates to 9

You can calculate A^2, B^2 and C^2 by using the coordinates of the endpoints of each line.

For example, B^2 is calculated as follows:



The endpoints, P1 and P2, have the coordinates (3, 2) and (6, 6).

The value B^2 is given by the formula:

$B^2 = (x^1 - x^2)^{\ 2} + (y^1 - y^2)^{\ 2}$

In this example:

$B^2 = (3 - 6)^2 + (2 - 6)^2$
$B^2 = (-3)^2 + (-4)^2$
$B^2 = 9 + 16$
$B^2 = 25$

A function IsRA() will:

- take three sets of integers as parameters representing the coordinates of the three endpoints that form a triangle
- return TRUE if the endpoints form a right-angled triangle, otherwise return FALSE.

In pseudocode, the operator '^' represents an exponent, which is the number of times a value is multiplied by itself. For example, the expression Value^2 may be written in pseudocode as Value ^ 2

Complete the pseudocode for the function IsRA().

```
FUNCTION IsRA(x1, y1, x2, y2, x3, y3 : INTEGER) RETURNS BOOLEAN
```

Answer

Mark Scheme and Guidance

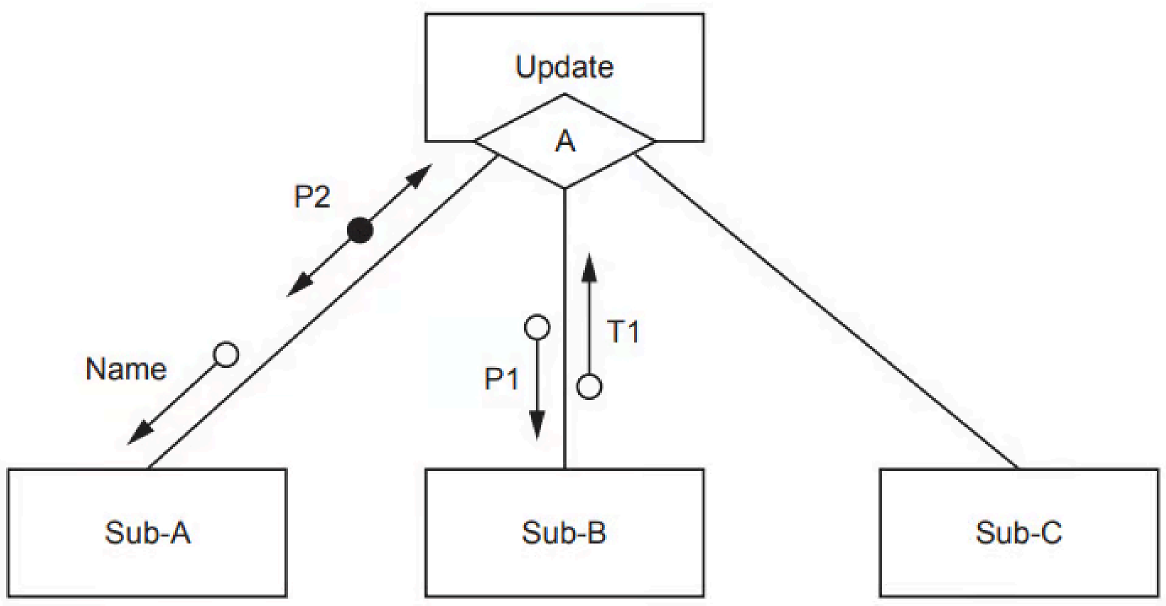
```
FUNCTION IsRA(x1, y1, x2, y2, x3, y3 : INTEGER) RETURNS
BOOLEAN
DECLARE Len1, Len2, Len3 : INTEGER

Len1 ← (x1 - x2) ^ 2 + (y1 - y2) ^ 2
Len2 ← (x1 - x3) ^ 2 + (y1 - y3) ^ 2
Len3 ← (x2 - x3) ^ 2 + (y2 - y3) ^ 2

IF (Len1 = Len2 + Len3) OR__
(Len2 = Len1 + Len3) OR__
(Len3 = Len1 + Len2) THEN
RETURN TRUE
ELSE
RETURN FALSE
ENDIF

ENDFUNCTION
```

Mark as follows:

1. Calculate the square of the length of at least one side
2. Calculation of all three lengths squared correctly
3. **One** correct comparison of square of three lengths
4. **All three** comparisons…
5. combined using logical operators / nested IF // completely correct selection
6. Return result correctly in both cases

(6 marks)

7 The structure chart illustrates part of the membership program:



Data item notes:

- Name contains the name of a club member
- P1 and T1 are of type real.

Write the pseudocode module headers for Sub-A and Sub-B.

Sub-A……………………………………………………………………………

Sub-B……………………………………………………………………………

Answer

Mark Scheme and Guidance

Means that Update calls (one of) either Sub-A, Sub-B or Sub-C

One mark for each point:

- reference to selection / decision / if

- naming all four modules correctly

(4 marks)

8 A teacher is designing a program to process pseudocode projects written by her students.

Each student project is stored in a text file.

The process is split into a number of stages. Each stage performs a different task and creates a new file named as shown:

File name	Comment
MichaelAday_src.txt	student project file produced by student Michael Aday
MichaelAday_S1.txt	file produced by stage 1
MichaelAday_S2.txt	file produced by stage 2

The teacher has defined the first program module as follows:

Module	Description
DeleteComment()	• called with a parameter of type string representing a line of pseudocode from a student's project file • returns the line after removing any comments Note on comments: A comment starts with two forward slash characters and includes all the remaining characters on the line. The following example shows a string before and after the comment has been removed: Before: IF X2 > 13 THEN //check if limit exceeded After: IF X2 > 13 THEN

Complete the pseudocode for module DeleteComment().

FUNCTION DeleteComment(Line : STRING) RETURNS STRING

Answer

Mark Scheme and Guidance

```
FUNCTION DeleteComment(Line : STRING) RETURNS STRING
DECLARE NewLine, TwoChars : STRING
DECLARE Count, TrimTo : INTEGER
CONSTANT Comment = "//"

NewLine ← Line
TrimTo ← 0
Count ← 1
WHILE Count < LENGTH(Line) AND TrimTo = 0
TwoChars ← MID(Line, Count, 2) // extract 2 chars
IF TwoChars = Comment THEN
```

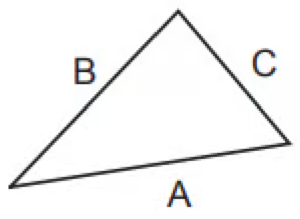
```
TrimTo ← Count
ENDIF
Count ← Count + 1
ENDWHILE

IF TrimTo <> 0 THEN
NewLine ← LEFT(Line, TrimTo - 1)
ENDIF

RETURN NewLine

ENDFUNCTION
```

Mark as follows:

1. Loop to length of Line (parameter) // length of Line -1
2. Terminate loop on first double slash
3. Attempt to extract one/two characters **in a loop**
4. Check for "//" after attempt at extraction **in a loop**
5. Record start position of comment // Calculate amount of trim required
6. Attempt to trim Line from start of comment
7. Completely correct trimmed Line from start of comment
8. Return Line after a reasonable overall attempt

(8 marks)

9 A triangle has sides of length A, B and C.



In this example, A is the length of the longest side.

This triangle is said to be right-angled if the following equation is true:

$A \times A = (B \times B) + (C \times C)$

A procedure will be written to check whether three lengths represent a right-angled triangle. The lengths will be input in any sequence.

The procedure IsRA() will:

- prompt and input three integer values representing the three lengths
- test whether the three lengths correspond to the sides of a right-angled triangle
- output a suitable message. The length of the longest side may not be the first value input.

The length of the longest side may **not** be the first value input.

Write pseudocode for the procedure IsRA().

Answer

Mark Scheme and Guidance

Example solution:

```
PROCEDURE IsRA()
DECLARE a, b, c : INTEGER

OUTPUT "Input length of the first side"
INPUT a
OUTPUT "Input length of the second side"
INPUT b
OUTPUT "Input length of the third side"
INPUT c

IF (a * a = (b * b) + (c * c)) OR__
(b* b = (a * a) + (c * c)) OR__
(c * c = (a * a) + (b * b)) THEN
OUTPUT "It is right-angled"
ELSE
OUTPUT "Not right-angled"
ENDIF
ENDPROCEDURE
```

Mark as follows:

1. Procedure heading **and** ending and declaration of all variables used
2. Appropriate prompt and input for each length
3. **One** correct length test
4. **All three** length tests // selection of which test is required
5. Output one of two messages following a reasonable attempt at MP3

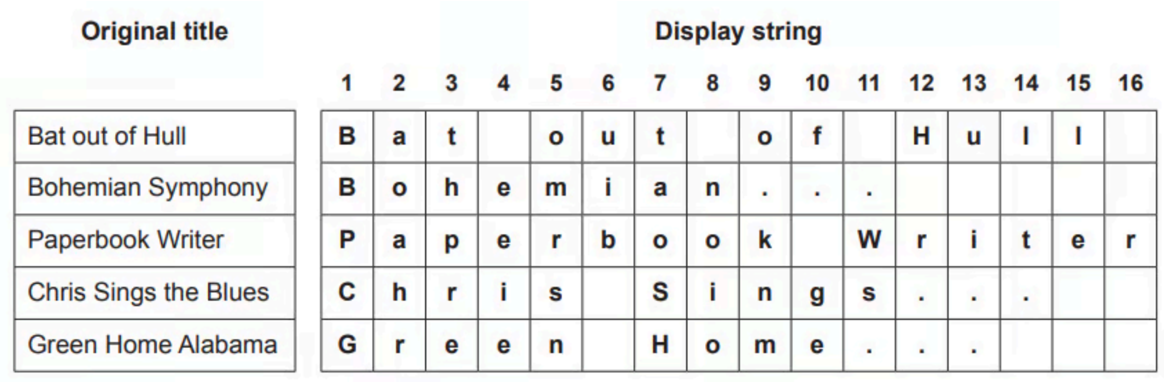
(4 marks)

10 A music player stores music in a digital form and has a display which shows the track being played.

Up to 16 characters can be displayed. Track titles longer than 16 characters will need to be trimmed as follows:

- Words must be removed from the end of the track title until the resulting title is less than 14 characters.
- When a word is removed, the space in front of that word is also removed.
- Three dots are added to the end of the last word displayed when one or more words have been removed.

The table below shows some examples:



Original title	Display string															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bat out of Hull	B	a	t		o	u	t		o	f		H	u	l	l	
Bohemian Symphony	B	o	h	e	m	i	a	n	.	.	.					
Paperbook Writer	P	a	p	e	r	b	o	o	k		W	r	i	t	e	r
Chris Sings the Blues	C	h	r	i	s		S	i	n	g	s	.	.	.		
Green Home Alabama	G	r	e	e	n		H	o	m	e	.	.	.			

A function Trim() will:

- take a string representing the original title
- return the string to be displayed.

Assume:

- Words in the original title are separated by a single space character.
- There are no spaces before the first word or after the last word of the original title.
- The first word of the original title is less than 14 characters.

Write pseudocode for the function Trim().

Answer

Mark Scheme and Guidance

Example solution:

```
Function Trim(Name : STRING) RETURNS STRING
CONSTANT Dots = "..."
CONSTANT Space = " "

IF LENGTH(Name) <= 16 THEN
RETURN Name
ENDIF

// Otherwise it has to be trimmed

WHILE LENGTH(Name) > 13
REPEAT
Name ← LEFT(Name, LENGTH(Name) - 1) // strip last char
UNTIL RIGHT(Name, 1) = Space // back to SPACE
ENDWHILE

Name ← LEFT(Name, LENGTH(Name) - 1) // remove the space
Name ← Name & Dots
RETURN Name

ENDFUNCTION
```

Mark as follows:

1. Function heading, ending, parameter and return type
2. If length of original string <= 16 then return original string

3. Any Conditional loop…
4. until string is short enough
5. Inner conditional loop / second condition to identify word(s) to remove from the end of string // Check to identify word(s) to remove from the end of string
6. Attempt to strip characters back to space
7. Correct removal of word/words from end of string and remove final space
8. Concatenate Dots and return result

Max 7 marks

(7 marks)

11 A teacher is designing a program to process pseudocode projects written by her students.

Each student project is stored in a text file.

The process is split into a number of stages. Each stage performs a different task and creates a new file.

For example:

File name	Comment
MichaelAday_src.txt	Student project file produced by student Michael Aday
MichaelAday_S1.txt	File produced by stage 1
MichaelAday_S2.txt	File produced by stage 2

The teacher has defined the first program module as follows:

Module	Description
DeleteSpaces()	• called with a parameter of type string representing a line of pseudocode from a student's project file • returns the line after removing any leading space characters The following example shows a string before and after the leading spaces have been removed: Before: " IF X2 > 13 THEN" After: "IF X2 > 13 THEN"

Answer

Mark Scheme and Guidance

Example algorithm based on finding position of first non-space character and then using substring function:

```
FUNCTION DeleteSpaces(Line : STRING) RETURNS STRING
DECLARE NewLine : STRING
DECLARE EndOfLeading : BOOLEAN
DECLARE Count, NumSpaces : INTEGER
DECLARE NextChar : CHAR
CONSTANT Space = " "

NumSpaces ← 0
EndOfLeading ← FALSE

FOR Count ← 1 TO LENGTH(Line)
NextChar ← MID(Line, Count, 1)
IF NextChar <> Space AND EndOfLeading = FALSE THEN
NumSpaces ← Count - 1 // the number to trim
EndOfLeading = TRUE
ENDIF
NEXT Count
```

```
NewLine ← RIGHT(Line, LENGTH(Line) - NumSpaces)

RETURN NewLine
ENDFUNCTION
```

Mark as follows:

1. Loop to length of parameter // Loop until first non-space character in Line
2. Extract a character **in a loop**
3. Identify first non-space character **in a loop**
4. Attempt at removing leading spaces in Line
5. Leading spaces removed from Line / / Create new string without leading space
6. Return a string following a reasonable attempt at removing leading spaces in Line

(6 marks)

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

A program uses many complex algorithms.

One algorithm is repeated in several places. The code for the algorithm is the same wherever it is used, but the calculations within the algorithm may operate on different data.

The result of each calculation is used by the code that follows it.

It is decided to modify the program and implement the algorithm as a separate module.

(i) State **two** benefits of this modification to the existing program.

[2]

(ii) Describe how the modification would be implemented.

[3]

Answer

Mark Scheme and Guidance

(i)

Two marks for the benefits:

1. The code can be called when needed
2. Any subsequent change to the algorithm / calculation needs to be made once only // Easier to manage / maintain (the program)
3. Less code / no code duplication
4. The algorithm / code / calculation can be designed / coded / tested once

Note: Max 2 marks

(ii)

1. One mark per point: Create a module / function / procedure / subroutine **using the (old) code / algorithm**

2. Replace the old code / algorithm wherever it appears with a call to the module / function / procedure / subroutine
3. Pass the data as parameter(s) // Use of global variable(s)
4. Return the result (of the calculation) // Assign result to global variable(s) / BYREF parameter(s)
5. Create local variables as needed (to perform the calculation)

Note: Max 3 marks

(2 marks)

13 A procedure TwoParts() will input a sequence of real values, one at a time.

The procedure will:

- process the sequence in two parts
- form a first total by adding the values until the first zero
- form a second total by adding the values after the first zero until the second zero
- output the average of the two totals, together with a suitable message.

Values input in the first part are totalled using global variable TotalA and those input in the second part are totalled using global variable TotalB.

Write pseudocode for the procedure TwoParts().

Answer

Mark Scheme and Guidance

Example solution:

```
PROCEDURE TwoParts()
DECLARE NextNum, Average : REAL

TotalA ← 0.0 // 0
TotalB ← 0.0 // 0
```

```
REPEAT
INPUT NextNum
TotalA ← TotalA + NextNum
UNTIL NextNum = 0

REPEAT
INPUT NextNum
TotalB ← TotalB + NextNum
UNTIL NextNum = 0

Average ← (TotalA + TotalB) / 2
OUTPUT "The average is ", Average
ENDPROCEDURE
```

Mark as follows:

1. Procedure heading and ending
2. Declare all local variables
3. Initialise TotalA and TotalB
4. First conditional loop until zero entered, summing TotalA // Loop until both parts (sequences) have been entered
5. Second conditional loop until zero entered, summing TotalB // Loop summing appropriate Totals
6. Calculation of average and output with a message // Calculation of the average for the values making up the two totals and both output with a suitable message

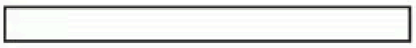
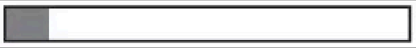
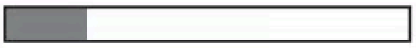
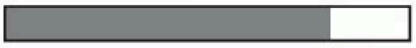
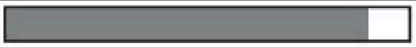
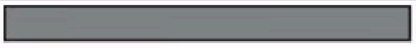
(6 marks)

14 (a) A program displays a progress bar to inform the user of the progress of tasks that take a significant time to complete, such as those involving file transfer operations.

Task progress is divided into 11 steps. Each step represents the amount of progress as a percentage. An image is associated with each step and each image is stored in a different file.

Different progress bar images may be selected. For a given image, files all have the same filename root, with a different suffix.

The table illustrates the process for using the image with filename root BargraphA

Step	Percentage progress	Image filename	Image
1	< 10	`BargraphA-1.bmp`	
2	>= 10 and < 20	`BargraphA-2.bmp`	
3	>= 20 and < 30	`BargraphA-3.bmp`	
9	>= 80 and < 90	`BargraphA-9.bmp`	
10	>= 90 and < 100	`BargraphA-10.bmp`	
11	100	`BargraphA-11.bmp`	

A procedure Progress() will:

- be called with two parameters:
  - an integer representing the percentage progress (0 to 100 inclusive)
  - a string representing the image filename root
- generate the full image filename
- call a procedure Display() using the full image filename as the parameter.

Write pseudocode for procedure Progress().

Answer

Mark Scheme and Guidance

Example Solutions:

```
PROCEDURE Progress(Percent : INTEGER, Root : STRING)
DECLARE StepValue : INTEGER
DECLARE Filename : STRING
```

```
StepValue ← (Percent DIV 10) + 1 // INT(Percent / 10) + 1
FileName ← Root & "-" & NUM_TO_STR(StepValue) & ".bmp"
CALL Display(Filename) ENDPROCEDURE
```

Alternative:

```
PROCEDURE Progress(Percent : INTEGER, Root : STRING)
DECLARE StepValue : INTEGER
DECLARE Filename : STRING
DECLARE Found : BOOLEAN
StepValue ← 1
Found ← FALSE
REPEAT
IF Percent < StepValue * 10 THEN
Found ← TRUE
ENDIF
StepValue ← StepValue + 1
UNTIL Found
Filename ← Root & "-" & NUM_TO_STR(StepValue - 1) & ".bmp"
CALL Display(Filename)
ENDPROCEDURE
```

Mark as follows for use of DIV/INT or loop solution:

1. Procedure heading, parameters and ending
2. Calculate StepValue as integer value // Attempt at calculating file number
3. Add 1 to obtain file number // Completely correct file number calculation
4. Use of NUM_TO_STR() to convert file number to string and use
5. Concatenate Root, hyphen, file number and ".bmp" suffix
6. Call Display() with filename as parameter following a reasonable attempt

(6 marks)

(b) The definition of procedure Progress() is provided here for reference

A procedure Progress() will:

- be called with two parameters:
    - an integer representing the percentage progress (0 to 100 inclusive)
    - a string representing the image filename root
- generate the full image filename
- call a procedure Display() using the full image filename as the parameter.

Progress() will be rewritten and a new module Progress2() produced with these requirements:

- an additional parameter of type integer will specify the total number of steps
- the image filename will be returned (procedure Display() will **not** be called from within Progress2()).

(i) Write pseudocode for the new module header.

[2]

(ii) State **one** benefit of increasing the number of steps.

[1]

Answer

Mark Scheme and Guidance

(i)

Example answer:

```
FUNCTION Progress2(Percent, Steps : INTEGER, Root : STRING) RETURNS STRING
```

- One mark for each: Keyword FUNCTION Progress2 **and** three parameters of correct type
- Keyword RETURNS and type string

(ii)

The progress display will more accurately show the progress of the task

(3 marks)

15 Seven program modules form part of a program. A description of the relationship between the modules is summarised below. Any return values are stated in the description.

Module name	Description
Mod-A	calls Mod-B followed by Mod-C
Mod-B	• called with parameters Par1 and Par2 • calls either Mod-D or Mod-E, determined when the program runs • returns a Boolean value
Mod-C	• called with parameters Par1 and Par3 • Par3 is passed by reference • repeatedly calls Mod-F followed by Mod-G
Mod-D	called with parameter Par2
Mod-E	• called with parameter Par3 • returns an integer value
Mod-F	called with parameter Par3
Mod-G	• called with parameter Par3 • Par3 is passed by reference

Parameters in the table are as follows:

- Par1 and Par3 are of type string.
- Par2 is of type integer.

(i) Identify the modules that would be implemented as functions.

[1]

(ii) Modules Mod-F and Mod-G are both called with Par3 as a parameter.
In the case of Mod-F, the parameter is passed by value.
In the case of Mod-G, the parameter is passed by reference.

Explain the effect of the **two** different ways of passing the parameter Par3.

[2]

Answer

Mark Scheme and Guidance

(i)

Mod-B() and Mod-E()

(ii)

Points required:

1. any change made to the parameter value / Par3 within Mod-G()is reflected in the (subsequent) value in the calling module / Mod-C() (after Mod-G() terminates) 2
2. any change made to the parameter value / Par3 within Mod-F()is NOT reflected in the (subsequent) value in the calling module / Mod-C() (after Mod-F() terminates)

Mark as follows:

1 mark for a reasonable attempt to explain

2 marks for full explanation including context

(3 marks)

16 A teacher is designing a program to process pseudocode projects written by her students.

The program analyses a student project and extracts information about each module that is defined (each procedure or function). This information is stored in a global 2D array ModInfo of type string.

A module header is the first line of a module definition and starts with either of the keywords PROCEDURE or FUNCTION.

An example of part of the array is given below. Row 10 of the array shows that a procedure header occurs on line 27 and row 11 shows that a function header occurs on line 35. "P" represents a procedure and "F" represents a function

	x = 1	**x = 2**	**x = 3**
ModInfo[10, x]	"27"	"P"	"MyProc(Z : CHAR)"
ModInfo[11, x]	"35"	"F"	"MyFun(Y : CHAR) RETURNS BOOLEAN"

The string stored in column 3 is called the module description. This is the module header **without** the keyword.

A valid module header will:

- be at least 13 characters long
- start with the keyword PROCEDURE or FUNCTION. The keyword may appear in either upper or lower case (or a mix of both) and **must** be followed by a space character.

The teacher has defined the first program module as follows:

Module	Description
Header()	• called with a parameter of type string representing a line of pseudocode • if the line is a valid procedure header, returns a string: "P<Module description>" • if the line is a valid function header, returns a string: "F<Module description>" • otherwise, returns an empty string For example, given the string: "FUNCTION Zap(X : INTEGER) RETURNS CHAR" Header()returns the string: "FZap(X : INTEGER) RETURNS CHAR"

Write pseudocode for module Header().

Answer

Mark Scheme and Guidance

Example:

```
FUNCTION Header(Line : STRING) RETURNS STRING

IF LENGTH(Line) >= 13 THEN
IF TO_UPPER(LEFT(Line, 9)) = "FUNCTION " THEN
RETURN "F" & RIGHT(Line, LENGTH(Line) - 9)
ENDIF
```

```
IF TO_UPPER(LEFT(Line, 10)) = "PROCEDURE " THEN
RETURN "P" & RIGHT(Line, LENGTH(Line) - 10)
ENDIF
ENDIF

RETURN ""

ENDFUNCTION
```

Mark as follows:

1. Function heading, parameter, return type and ending
2. Check that the line is at least 13 characters long before attempting to extract and return empty string
3. Attempt at: Extract characters, 9 or 10 characters, corresponding to keyword plus space and compare with appropriate keyword plus space
4. Completely correct MP3
5. Use of type case conversion to allow for 'any case'
6. Calculation of 'rest of line' and concatenation with 'P or 'F'
7. Return string

(7 marks)

17 A global array is declared in pseudocode as follows:

```
DECLARE Data : ARRAY[1:150] OF STRING
```

A function TooMany() will:

1. take two parameters:

   • a string (the search string)

   • an integer (the maximum value)

2. count the number of strings in the array that exactly match the search string

3. return TRUE if the count is greater than the maximum value, otherwise will return FALSE

Write pseudocode for the function TooMany().

Answer

Mark Scheme and Guidance

```
FUNCTION TooMany(Search : STRING, Max : INTEGER) RETURNS BOOLEAN
DECLARE Count, Index : INTEGER

Count ← 0

FOR Index ← 1 TO 150
IF Data[Index] = Search THEN
Count ← Count + 1
ENDIF
NEXT Index

IF Count > Max THEN
RETURN TRUE
ELSE
RETURN FALSE
ENDIF
ENDFUNCTION
```

MP1 Function heading, ending and return type
MP2 Declare Count and Index as integers
MP3 Initialise Count
MP4 Loop (any type) for 150 iterations
MP5 Compare Data element with parameter - if equal, increment Count in a loop
MP6 Compare Count with Max and return Boolean in both cases outside the loop

(6 marks)

18 (a) The pseudocode OUTPUT command starts each output on a new line.

A new procedure MyOutput() will take a string and a Boolean parameter.
MyOutput() may be called repeatedly and will use concatenation to build a string using a global variable MyString, up to a maximum length of 255 characters.

MyString will be output in either of these two cases:

1. The Boolean parameter value is TRUE

2. The resulting string (after concatenation) would be longer than 255 characters.

If MyString is not output, the string is concatenated with MyString.

For example, the calls to MyOutput() given below would result in the output as shown:

```
MyOutput("Hello ", FALSE)
MyOutput("ginger ", FALSE)
MyOutput("cat", TRUE)
MyOutput("How are you?", TRUE)
```

Resulting output:

```
Hello ginger cat
How are you?
```

Notes:

- MyString is initialised to an empty string before MyOutput() is called for the first time.
- No string passed to MyOutput() will be longer than 255 characters.

Write pseudocode for MyOutput().

Answer

Mark Scheme and Guidance

```
PROCEDURE MyOutput(NewString : STRING, EOL : BOOLEAN)

IF LENGTH(MyString) + LENGTH(NewString) > 255 THEN
OUTPUT MyString // Resulting string would be too long
MyString ← NewString
ELSE
MyString ← MyString & NewString // Concat with MyString
IF EOL = TRUE THEN
OUTPUT MyString
MyString ← ""
ENDIF
ENDIF ENDPROCEDURE
```

MP1 Procedure heading, including parameters, and ending
MP2 Produce concatenated string
MP3 ... Check whether resulting string would be too long

MP4 If so, then output old MyString
MP5 ... and assign NewString to MyString
MP6 Else concatenate NewString to MyString

MP7 (test for length < 255) Test EOL – If TRUE then Output
MP8 ... and reset MyString to empty string

(7 marks)

(b) The design of the procedure given in **part (a)** is modified and MyString is changed from a global to a local variable declared in MyOutput().

When the modified procedure is converted into program code, it does not work as expected.

Explain why it does not work as expected.

Answer

Mark Scheme and Guidance

MP1 A new (instance of) variable MyString is created each time the procedure is called / executed
MP2 So the previous contents are lost

(2 marks)

19 A new module has been defined:

Module	Description
GetField()	• takes two parameters: ◦ a string containing a message ◦ an integer containing a field number • If the field number is valid (in the range 1 to 3, inclusive), it returns a string containing the required field, otherwise it returns an empty string.

As a reminder, a message is defined as follows:

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

Field number	Field name	Description
Not applicable	STX	a single character marking the start of the message (ASCII value 02)
1	DestinationID	three numeric characters that identify the destination computer
2	SourceID	three numeric characters that identify the source computer
3	Data	a variable length string containing the data being sent (Minimum length is 1 character)
Not applicable	ETX	a single character marking the end of the message (ASCII value 03)

Write pseudocode for module GetField().

Answer

Mark Scheme and Guidance

```
FUNCTION GetField(Msg : STRING, FieldNo : INTEGER) RETURNS STRING
DECLARE RetString : STRING

CASE OF FieldNo
1 : RetString ← MID(Msg, 2, 3)
2 : RetString ← MID(Msg, 5, 3)
3 : RetString ← MID(Msg, 8, LENGTH(Msg) - 8)
OTHERWISE : RetString ← ""
ENDCASE

RETURN RetString
ENDFUNCTION
```

MP1 Use of CASE … ENDCASE or IF … THEN … ENDIF
MP2 Field 1 and Field 2 extracted correctly
MP3 Calculate a length of field 3
MP4 Field 3 extracted correctly
MP5 Return empty string in case of invalid parameter (via OTHERWISE or initialisation)
MP6 Final RETURN, after a reasonable attempt

(6 marks)

20 A procedure Count() will:

1. input a value (all values will be positive integers)
2. count the number of odd values and count the number of even values
3. repeat from step 1 until the value input is 99
4. output the two count values, with a suitable message.

The value 99 must not be counted.

Write pseudocode for the procedure Count().

Answer

Mark Scheme and Guidance

Example Solution

```
PROCEDURE Count()
DECLARE COdd, CEven, ThisNum : INTEGER

COdd ← 0
CEven ← 0

INPUT ThisNum

WHILE ThisNum <> 99
IF ThisNum MOD 2 = 1 THEN

COdd ← COdd + 1
ELSE
CEven ← CEven + 1
ENDIF
INPUT ThisNum
ENDWHILE

OUTPUT "Count of odd and even numbers: ", COdd, CEven
ENDPROCEDURE
```

Mark as follows **Max 6** marks:

1. Procedure heading and ending
2. **Local** COdd, CEven and ThisNum declared as integers
3. Conditional loop while ThisNum <> 99
4. Input ThisNum **in a loop**
5. Check ThisNum is odd or even **in a loop**
6. Increment appropriate count **in a loop**, both counts must have been initialised **before loop**
7. **After the loop** output COdd and CEven **with a** suitable message following a reasonable attempt at counting

(6 marks)

21 A procedure CreateFiles() will take two parameters:

- a string representing a file name
- an integer representing the number of files to be created.

The procedure will create the number of text files specified.

Each file is given a different name. Each file name is formed by concatenating the file name with a suffix based on the file number. The suffix is always three characters.

For example, the call CreateFiles("TestData", 3) would result in the creation of the three files, TestData.001, TestData.002 and TestData.003.

Each file will contain a single line. For example, file TestData.002 would contain the string:

This is File TestData.002

A module CheckFiles() will count the number of files produced by CreateFiles() in part **(a)**.

CheckFiles() will take a string representing a file name and return the number of files found.

(i) Identify the type of module that should be used for CheckFiles().

[1]

(ii) Write the module header for CheckFiles().

[1]

Answer

Mark Scheme and Guidance

(i)

Function

(ii)

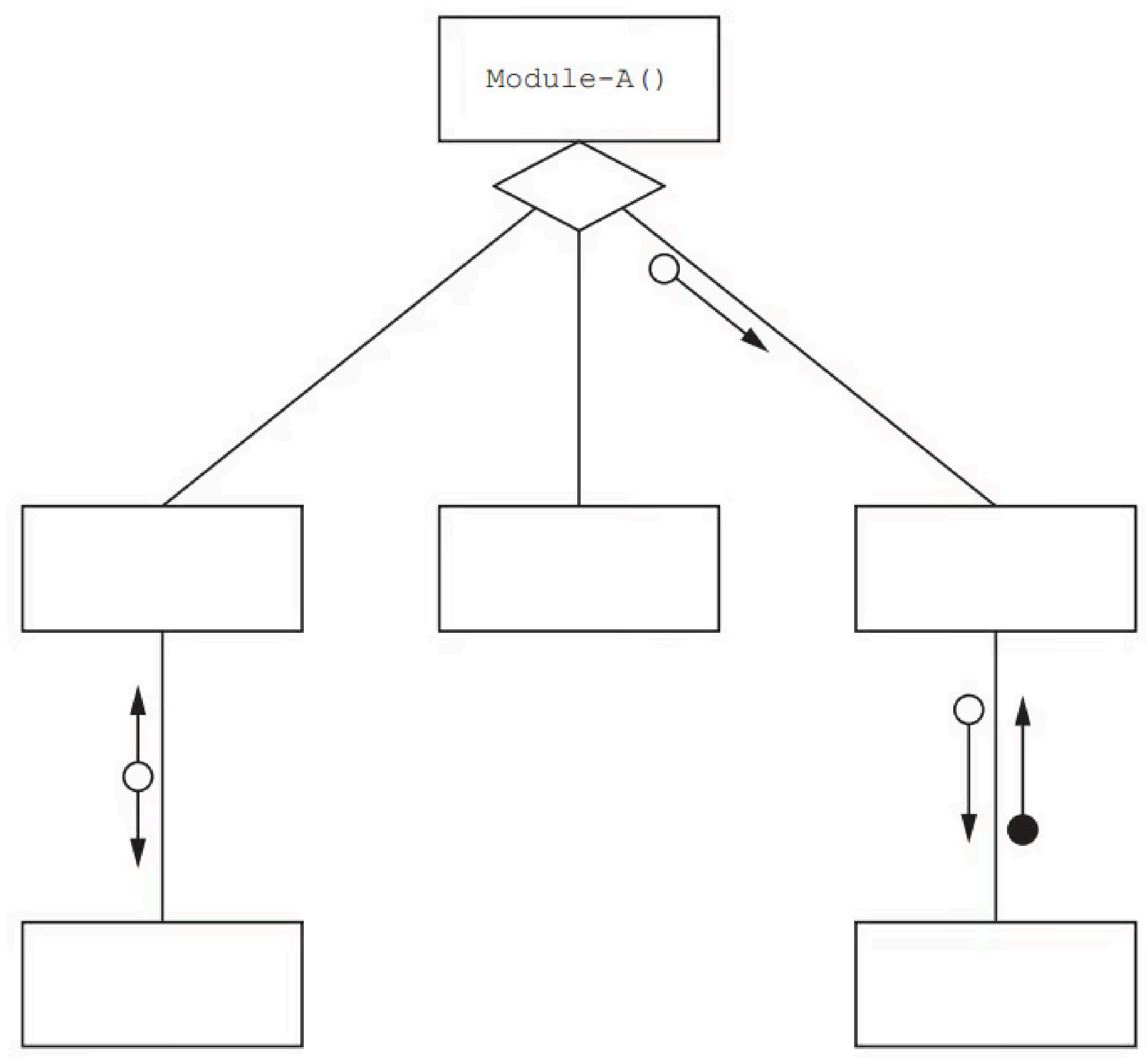
```
FUNCTION CheckFiles(NameRoot : STRING) RETURNS INTEGER
```

(2 marks)

22 A program contains six modules:

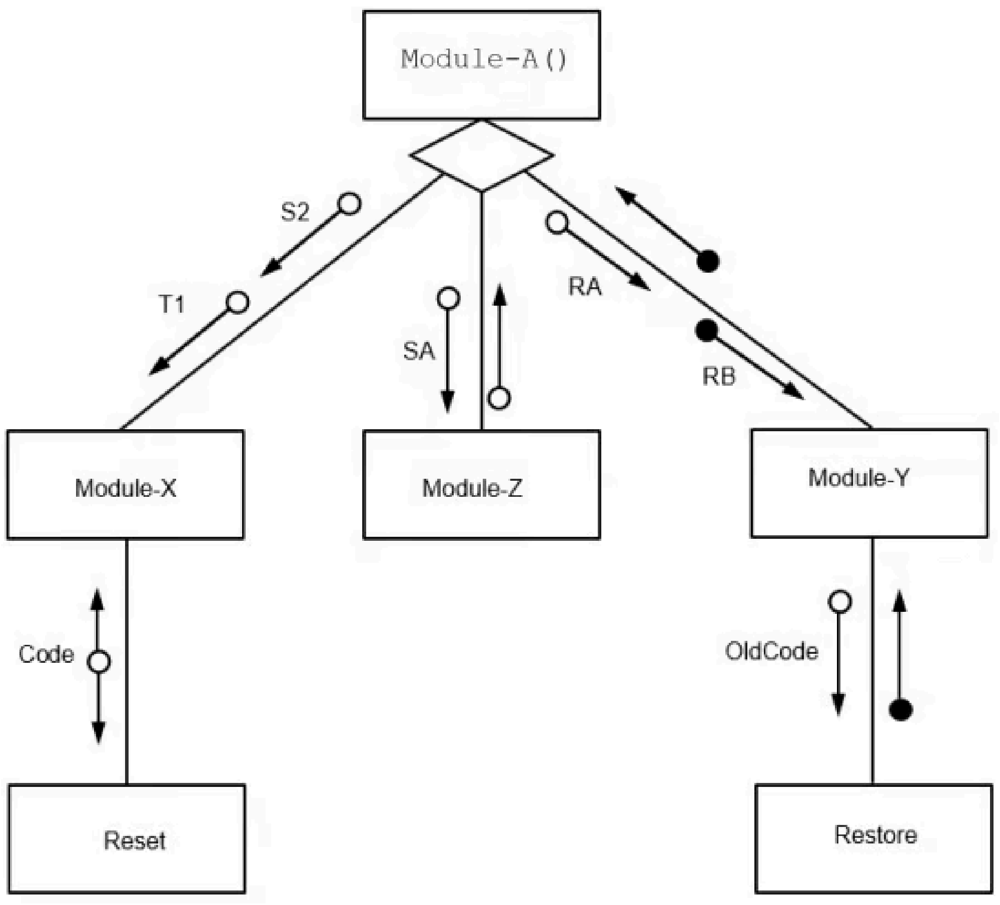
Pseudocode module header
PROCEDURE Module-A()
PROCEDURE Module-X(T1 : INTEGER, S2 : REAL)
PROCEDURE Reset(BYREF Code : INTEGER)
FUNCTION Restore(OldCode : INTEGER) RETURNS BOOLEAN
FUNCTION Module-Z(SA : INTEGER) RETURNS INTEGER

Module-X() calls Reset()
Module-Y() calls Restore()



Answer

Mark Scheme and Guidance



One mark per point:

1. Module names
2. Parameters with labels to Module-X and between Module-X and Reset
3. Parameter (with label) to Module-Z and return from Module-Z
4. Parameters with labels to Module-Y and Restore and return values from Module-Y

(4 marks)

23 Copies of the same program will run on each computer. The program contains a global variable MyID of type string, which contains the unique ID of the computer in which the program is running.

When messages are received, they are placed on one of two stacks. Stack 1 is used for messages that have reached their destination and stack 2 is used for messages that will be forwarded on to another computer.

Additional modules are defined:

Module	Description
StackMsg() (already written)	• takes two parameters: ○ a string representing a message ○ an integer representing the stack number • adds the message to the required stack • returns TRUE if the message is added to the required stack, otherwise returns FALSE
ProcessMsg()	• takes a message as a parameter of type string • ignores any message with a zero-length data field • extract the DestinationID from the message • checks whether the DestinationID is this computer or whether the message is to be forwarded • uses StackMsg() to add the message to the appropriate stack • outputs an error if the message could not be added to the stack

Write pseudocode for module ProcessMsg().

Module StackMsg() must be used.

Answer

Mark Scheme and Guidance

Example solution

```
PROCEDURE ProcessMsg(ThisMsg : STRING)
DECLARE ThisDest : STRING
DECLARE Response : BOOLEAN
DECLARE StackNum : INTEGER

IF LENGTH(ThisMsg) >= 4 THEN

ThisDest ← LEFT(ThisMsg, 3)

IF ThisDest = MyID THEN // It's for this computer
StackNum ← 1
ELSE
StackNum ← 2
ENDIF

Response ← StackMsg(ThisMsg, StackNum)

IF Response = FALSE THEN
OUTPUT "Message discarded - no room on stack"
ENDIF

ENDIF
ENDPROCEDURE
```

Mark as follows:

1. Ignore message if data field is empty
2. Extract ThisDest from ThisMsg
3. Test if destination is this computer
4. Attempt to use StackMsg()
5. **Fully correct** use of StackMsg()for **both** cases / stacks
6. Test StackMsg() return value for **both** cases
7. Following a reasonable attempt at MP6 output warning if StackMsg() returns FALSE

(7 marks)

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

Write pseudocode for procedure RandList().

Answer

Mark Scheme and Guidance

```
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
```

MP1 Procedure heading and ending
MP2 Local loop counter Count as integer
MP3 Loop to iterate 25 times or more for each unique number
MP4 'Attempt' to generate a random number including use of INT() **in a loop**
MP5 Ensure that number generated is greater than previous and change 'previous'
MP6 Output random number after an attempt at MP5 **in a loop**

(6 marks)

25 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.

Answer

Mark Scheme and Guidance

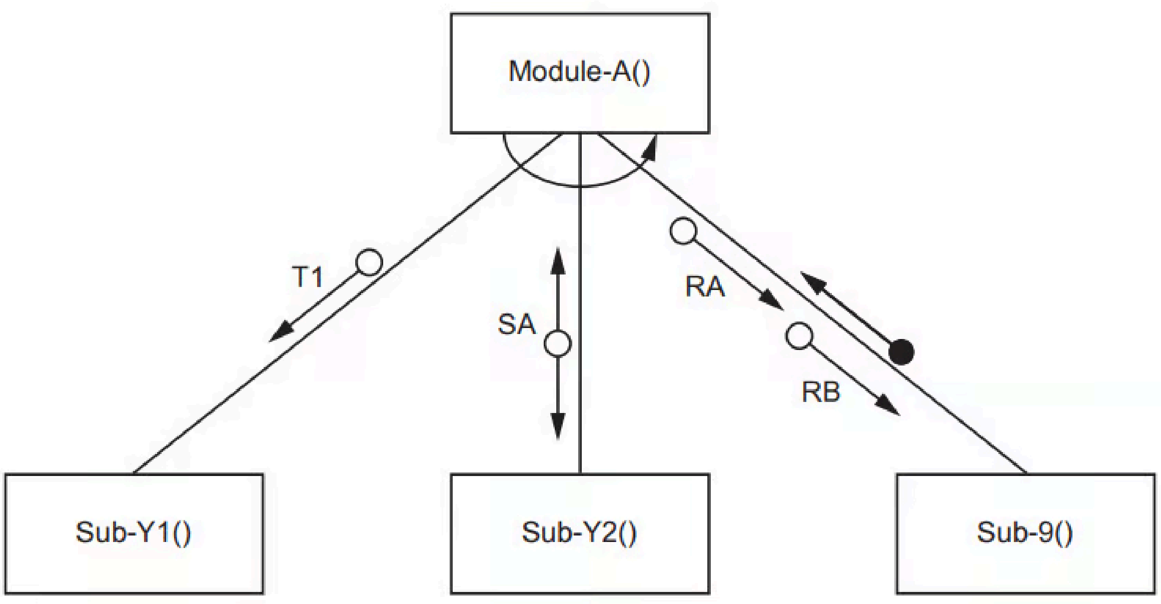
```
FUNCTION TestNum(ThisNum : STRING) RETURNS INTEGER

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
```

MP1 Function heading and ending **including** parameter and return type
MP2 Test for Condition 1
MP3 Test for Condition 2
MP4 Test for Condition 3
MP5 Return the highest value if more than one condition is satisfied
MP6 Return zero if no condition matched

(6 marks)

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



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.

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().

Answer

Mark Scheme and Guidance

PROCEDURE Sub-9(BYREF Param : MyType)

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

(2 marks)

27 (a) 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>"45420Hello 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

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.

Answer

Mark Scheme and Guidance

```
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
```

Conditional loop
MP1 Conditional loop
MP2 Test for terminator in both cases
MP3 Use GetData() to get the data from the message
MP4 OUTPUT the data **in a loop**
MP5 INPUT the data reply
MP6 'Attempted ' use of Transmit to send it **in a loop**
MP7 Correct formation of parameters to Transmit()

(7 marks)

(b) 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..

Modification...

Answer

Mark Scheme and Guidance

Note: Max 3 marks **(from either limitation or modification list)**

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 marks)