Full Chapter Worksheet / Chapter 8 – Answers

November 2024 (21)

(b)

One mark for each point

- `OPENFILE MyPassword.txt FOR WRITE`
- `WRITEFILE MyPassword.txt, Password`
- `CLOSEFILE MyPassword.txt`

(c)

One mark for each point

- needs to be retrieved on demand // saved for a later date
- storage must be non-volatile

SP 2023 / 2B

4(a)	**One** mark for each item correctly circled • open • write
4(b)	**One** mark for each correct point **max two**: • before trying to open the file • check the file exists • if the file does not exist, then output a suitable error message.

November 2024 (22)

10

One mark per mark point (**max six**)

- Declaration of appropriate string variable(s)
- Input of a line of text
- Correct use of `LCASE`
- Correct use of `LENGTH`
- Opening of at least **one** of the required text files using given names (`Main.txt`, `Lowercase.txt`)
- Storing of correct line of text to `Main.txt`
- Storing of correct lines to `Lowercase.txt`
- Closing of both text files
- Correct output

For example:

```
DECLARE LineOfText : STRING
DECLARE LowercaseText: STRING
INPUT LineOfText
OPENFILE Main.txt FOR WRITE
WRITEFILE Main.txt, LineOfText
CLOSEFILE Main.txt
LowercaseText ← LCASE(LineOfText)
OUTPUT LowercaseText, LENGTH(LineOfText)
OPENFILE Lowercase.txt FOR WRITE
WRITEFILE Lowercase.txt, LowercaseText
CLOSEFILE Lowercase.txt
```

June 2024 (21)

7(a)

One mark per mark point, **max two**

- Data stored in a file gives a permanent copy / prevents the data from being lost
- … so it can be recalled / used again in a program / in another program
- can be stored elsewhere / transferred to another computer

7(b)

One mark per mark point, **max four**

MP1 Declaration of a string variable
MP2 Correct use of `OPENFILE` keywords for reading
MP3 Correct use `READFILE` with string variable
MP4 Correct use of `UCASE` and `LENGTH` functions **and** output of each
MP5 Correct use of `CLOSEFILE`

Example:

```
DECLARE Words : STRING
OPENFILE Quotation.txt FOR READ
READFILE Quotation.txt, Words
OUTPUT UCASE(Words), LENGTH(Words)
CLOSEFILE Quotation.txt
```

SP 2023 / 2B

10(a)

```
P   Computer Science
Q   16
R   Science
S   7
T   Sci
```

10(b)

One mark correct function assigned to `F` **one** mark correct parameters

```
F ← SUBSTRING(P,1,8)
```

June 2024 (22)

4(a)

One mark per mark point

MP1 length check …
MP2 … to ensure the product code entered is 6 characters in length
MP3 format check …
MP4 … to ensure the first two characters of the product code entered are "PD"
MP5 range check …
MP6 … to ensure that the value of the last four figures of the product code entered is between 1000 and 9999

4(b)(i)

One mark for correct use of `LENGTH` operation, **one** mark for appropriate test

Example:

```
REPEAT
   INPUT Product
UNTIL LENGTH(Product) = 6
```

4(b)(ii)	**One** mark for correct use of `SUBSTRING` operation, **one** mark for appropriate test
	Example:
	`REPEAT`
	`  INPUT Product`
	`UNTIL SUBSTRING(Product, 1, 2) = "PD"`

June 2024 (23)

7(a)	**One** mark per mark point
	
	MP1 Assignment of given string to `FullText`
	MP2 Correct use of `SUBSTRING` and assignment of reduced string to own variable
	MP3 Correct use of `UCASE`
	MP4 Output of both strings
	
	Example:
	
	`FullText ← "IGCSE Computer Science at Cambridge"`
	`PartText ← SUBSTRING(FullText, 7, 16)`
	`OUTPUT PartText, UCASE(FullText)`

7(b)	**One** mark per mark point
	
	MP1 Opening the correct text file for writing
	MP2 Writing the variable from **part (a)** to the file
	MP3 Closing the text file after writing
	
	Example:
	
	`OPENFILE "Subjects.txt" FOR WRITE`
	`WRITEFILE "Subjects.txt", PartText`
	`CLOSEFILE "Subjects.txt"`

June 2025 (21)

4(a)

One mark per bullet point. **Max** 5

- Correct input of number
- Use of substring ( )
- …to extract the correct number
- Correct use of IF statements/Case statement with matching `END IF` / `END CASE`
- Output of matching locations
- Output of unknown

Example

```
OUTPUT "Please enter a telephone number "
INPUT Telephone
Value ← SUBSTRING(Telephone, 2, 2)
IF Value = "44"
  THEN
    OUTPUT "UK"
  ELSE
    IF Value = "20"
      THEN
        OUTPUT "Egypt"
      ELSE
        OUTPUT "Unknown"
    ENDIF
ENDIF
```

4(b)

One mark per bullet point. **Max** 3

- Find length of input.
- Use a suitable loop (repeat/while).
- Using the correct condition
- Prompt for a 13-digit telephone number and input number inside loop.

June 2023 (21)

2	A

4

One mark per mark point, max **four**

`DIV`, max **two**

- To perform integer division
- Meaning only the whole number part of the answer is retained
- Example of `DIV` For example `DIV(9,4) = 2`

`ROUND`, max **two**

- To return a value rounded to a specified number of digits / decimal places
- The result will either be rounded to the next highest or the next lowest value
- … depending on whether the value of the preceding digit is >=5 or <5
- Example of `ROUND` for example, `ROUND(4.56, 1) = 4.6`

June 2023 (23)

2	**One** mark per mark point, max **four** `MOD`, max **two** • To perform (integer) division when one number is divided by another • … and find the remainder • Allow example e.g. `7 MOD 2 = 1` `RANDOM`, max **two** • To generate (pseudo) random numbers • …(usually) within a specified range • Allow example e.g. `RANDOM() * 10` returns a random number between `0` and `10`

Practice Questions

1.

```
1. DECLARE Items, BoxCapacity : REAL
2. DECLARE FullBox, RemainItems : INTEGER
3.
4. OUTPUT "Enter the total number of items:"
5. INPUT Items
6. OUTPUT "Enter the box capacity:"
7. INPUT BoxCapacity
8.
9. FullBox ← DIV(Items,BoxCapacity)
10.RemainItems ← MOD(Items,BoxCapacity)
11.
12.OUTPUT "You can fill ", FullBox, " full boxes."
13.OUTPUT "Remaining items: ", RemainItems
```

2.

```
4. DECLARE Dice1, Dice2, Sum : INTEGER
5.
6. Dice1 ← ROUND(((RANDOM()*5)+1),0)
7. Dice2 ← ROUND(((RANDOM()*5)+1),0)
8. Sum ← Dice1 + Dice2
9. OUTPUT "Dice 1: ", Dice1
10.OUTPUT "Dice 2: ", Dice2
11.OUTPUT "Sum: ", Sum
```

November 2024 (23)

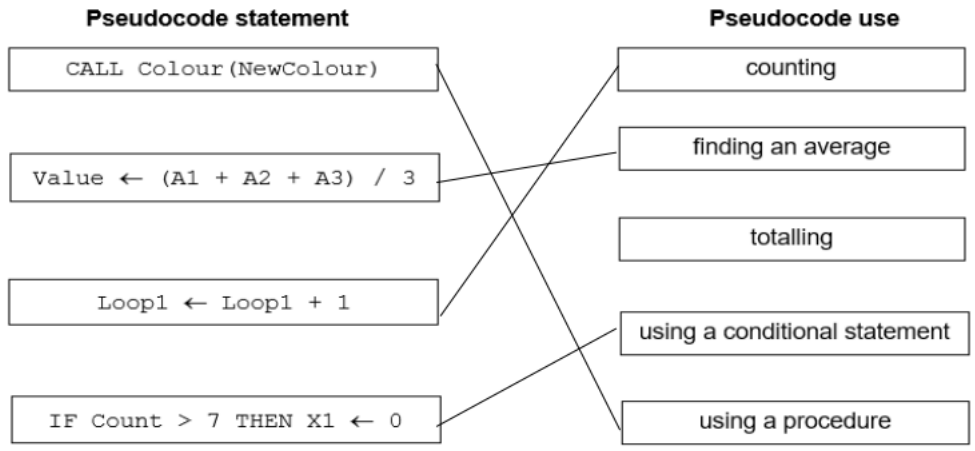
6(a)	**One** mark for each point • `06 C ← 1` • `08 W ← W + A[C]` • `11 X ← W / (C - 1) // ROUND(W / (C - 1),0)`

6(b)	**One** mark for outputting `X`
	One mark for outputting `C - 1`
	One mark for suitable messages
	
	Example:
	
	`12 OUTPUT "Number of values stored in the array is ", C - 1`
	`13 OUTPUT "Average of non-zero elements in the array is ", X`

6(c)	**One** mark for meaningful identifier for the array `A` **`Values`** **Two** marks for 3 meaningful identifiers for the variables **or** **One** mark for 1 to 2 meaningful identifiers for the variables `C` **`Index`** `X` **`Average`** `W` **`Total`**

November 2023 / 22

5	**One** mark per mark point, max **three** MP1 variables and constants should have meaningful identifiers MP2 …so that programmers/future programmers are able to understand their purpose MP3 they are **both** used for data storage MP4 constants store values that never change during the execution of a program // by example MP5 variables contain values that have been calculated within the program / can change during the execution of the program // by example

3(a)	**One** mark for each correct line. 

3(b)

One mark per mark point, max **four**

MP1 initialise a variable to store the lowest number set to a high value (at least 100) / first value in the array
MP2 loop structure to iterate 25 times
MP3 use of `IF` to check if the array element is less than the current lowest value
MP4 …if it is, set this to the lowest value
MP5 output the result (with an appropriate message) after the loop

Example answer:

```
Min ← 100
FOR Count ← 1 TO 25
    IF Temperatures[Count] < Min
      THEN
        Min ← Temperatures[Count]
    ENDIF
NEXT Count
OUTPUT "The lowest temperature is ", Min
```

June 2024 (21)

5

One mark per mark point

MP1 Correct input statement with appropriate variable
MP2 Elements of selection statement present – `CASE OF ENDCASE`
MP3 At least one correct branch in the case statement
MP4 All branches from 1 to 4 correct
MP5 Correct use of `OTHERWISE` with correct output.

For example:

```
INPUT Number
CASE OF Number
  1 : OUTPUT Number
  2 : OUTPUT Number
  3 : OUTPUT Number
  4 : OUTPUT Number
  OTHERWISE OUTPUT "ERROR"
ENDCASE
```

Or

```
INPUT Number
CASE OF Number
  1 : OUTPUT 1
  2 : OUTPUT 2
  3 : OUTPUT 3
  4 : OUTPUT 4
  OTHERWISE OUTPUT "ERROR"
ENDCASE
```

SP 2023 / 2B

12(a)	The whole algorithm must be rewritten for full marks.
	
	One mark for each of the following:
	• initialising counter outside the loop
	• updating counter inside loop
	• suitable exit value at start of loop
	• correct use of `WHILE … DO … ENDWHILE`
	
	Example:
	`B ← FALSE`
	`INPUT Num`
	`Counter ← 1`
	`WHILE Counter <= 12 DO`
	`  IF A[Counter] = Num`
	`  THEN`
	`  B ← TRUE`
	`  ENDIF`
	`  Counter ← Counter + 1`
	`ENDWHILE`

12(b)	Linear search
12(c)	Any **three** from: • `WHILE` has criteria check at start // pre-condition • code inside `WHILE` may never run • `REPEAT UNTIL` has criteria check at end // post-condition • `REPEAT UNTIL` will always run at least once

June 2025 (21)

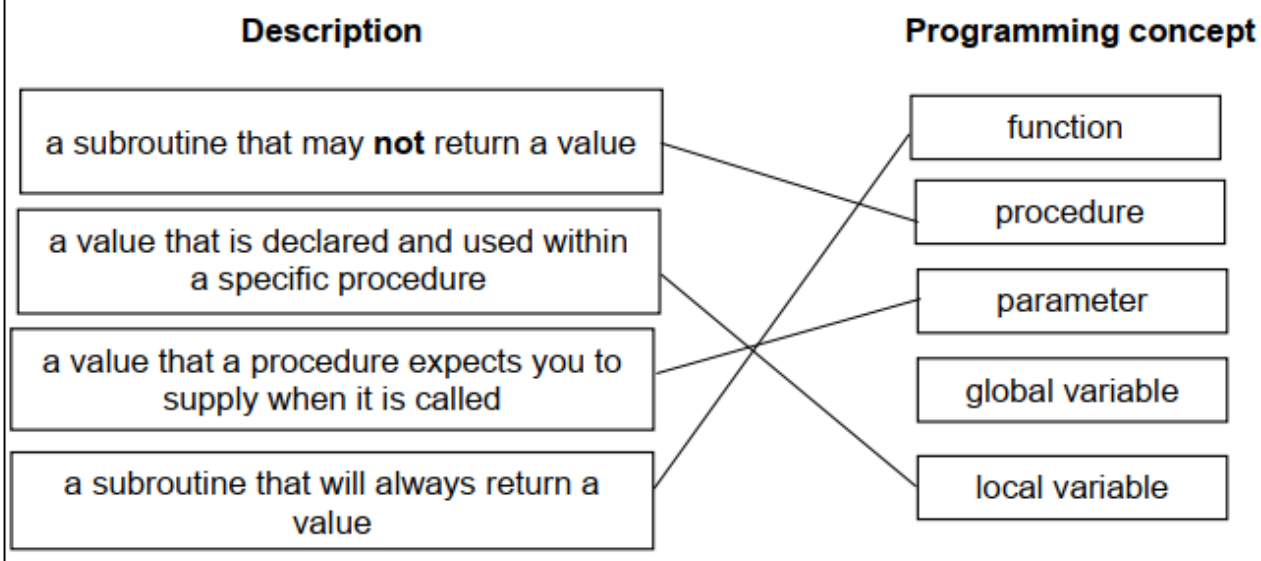
3	**One** mark per bullet point
	
	• Initialising a count variable outside of loop
	• Correct use of loop with condition for 10 times (`WHILE (DO)...ENDWHILE, REPEAT...UNTIL`) with close
	• Incrementing the count variable inside the loop
	• Rest of algorithm correct
	
	Examples
	
	`Count ← 1`
	`REPEAT`
	`  OUTPUT "Please enter a name"`
	`  INPUT Name`
	`  Names[Count] ←Name`
	`  Count ← Count + 1`
	`UNTIL Count > 10`
	
	Or
	
	`Count ← 1`
	`WHILE Count <= 10 DO`
	`  OUTPUT "Please enter a name"`
	`  INPUT Name`
	`  Names[Count] ←Name`
	`  Count ← Count + 1`
	`ENDWHILE`

November 2023 (23)

7	**one** mark for first description **one** mark for matching difference **max four** • local variables - scope is a defined block of code/subroutine/procedure/function • global variables – scope is the whole program • local variables - value cannot be changed elsewhere in the program • global variables – value can be changed anywhere in the program

June 2024 (21)

2(a) **One** mark for each correct line



2(b) **One** mark per mark point

- Correct key word - `CALL`
- Procedure name with correct parameters - `Average(25, 50)`

```
CALL Average(25, 50)
```

2(c) **One** mark per mark point, **max three**

- Procedures / functions divide the program into smaller manageable segments
- … making it more readable / easier to understand / easier to debug
- Procedures / functions with meaningful names help to provide documentation for the program / enable/help/provide abstraction
- Procedures and functions may be re-used (in the program / in other programs / as part of a library) / run from a single line
- Procedures and functions can reduce / eliminate (repeated) code

March 2023 (22)

10(a)	**One** mark for each correct line
	`DECLARE X : INTEGER`
	`DECLARE Y : REAL`
	`DECLARE Z : BOOLEAN`

10(b)	**One** mark for using `FUNCTION` and `ENDFUNCTION` and `RETURNS BOOLEAN`
	One mark for naming the function `Same`
	One mark for defining the two parameters correctly
	One mark for comparing the two parameters using `ROUND`
	One mark for correctly returning `TRUE` and `FALSE`
	One mark for correct function call
	
	Example definition:
	`FUNCTION Same(A : INTEGER, B : REAL) RETURNS BOOLEAN`
	`  IF A = ROUND(B,0)`
	`  THEN`
	`  RETURN TRUE`
	`  ELSE`
	`  RETURN FALSE`
	`  ENDIF`
	`ENDFUNCTION`
	
	Example call:
	`Z ← Same(X,Y)`

10(c)	A function is defined once **and** called many times **or** Define – setting up the function **and** call is using a function

June 2023 / 23

3	**One** mark per mark point, max **three** MP1 A **call statement** is used in order to make use of a function // the function is called **using its identifier** MP2 Parameters are / may be passed (from the main program) to the function (to be used within the function) MP3 The function performs its task … MP4 … and returns a value / values to the main program

NOVEMBER 2023 / 22

8	**One** mark per mark point, max **four** Procedures, max **three** MP1 to enable the programmer to write a collection of programming statements under a single identifier MP2 to allow modular programs to be created // to allow procedures to be re-used within the program or in other programs MP3 to make program creation faster because procedures can be re-used // to enable different programmers to work on different procedures in the same project MP4 to make programs shorter (than using the repeated code) / using less duplication of code // to make programs easier to maintain due to being shorter. Parameters, max **three** MP5 to pass values from the main program to a procedure / function MP6 …so that they can be used in the procedure / function MP7 allow the procedure / function to be re-used **with different data**.

Practice Questions

1.

```
PROCEDURE GreetUser(Name: STRING)
    OUTPUT "Hello, ", Name, "! Welcome to the program."
END PROCEDURE


CALL GreetUser("Alice")
```

2.

```
PROCEDURE CalculatePerimeter(Length: INTEGER, Width: INTEGER)
    Perimeter ← 2 * (Length + Width)
    OUTPUT "The perimeter is ", Perimeter
END PROCEDURE


CALL CalculatePerimeter(8, 6)
```

March 2024 (22)

6

One mark for identifying a type of iteration, **one** mark for accompanying description **max four**

- count controlled (1) number of iterations is pre-determined (1)
- pre-condition (1) checks condition at start of loop // loop may not iterate (1)
- post-condition (1) checks condition at end of loop // loop always iterates at least once (1)

10

One mark for each technique
One mark for a matching description
max six

- use comments …
- … to explain the purpose of each section of code
- …for example, logic / syntax

- use meaningful identifier names to …
- … clearly identify the purpose
- … of variables, constants, arrays, procedures etc
- … by example

- use procedures and functions …
- … to avoid repeated code
- … simplify logic

- use indentation and white space …
- … to make the program readable