Work Sheet-Structured Programming 20191010

Q1

2 **(a)** **(i)** Procedures and functions are examples of subroutines.

State a reason for using subroutines in the construction of an algorithm.

.....

..... [1]

(ii) Give **three** advantages of using subroutines in a program.

1 .....

.....

2 .....

.....

3 .....

.....

[3]

(iii) The following pseudocode uses the subroutine `DoSomething()`.

```
Answer ← 23 + DoSomething("Yellow")
```

State whether the subroutine is a function or a procedure. Justify your answer.

Type of subroutine .....

Justification .....

.....

[2]

(b) Stepwise refinement is often used in the development of an algorithm.

Describe **stepwise refinement**.

.....

.....

.....

.....[2]

Q2

3 In a chemical factory, a procedure, `CheckSensor()` is required to allow an operator to monitor the temperature in different locations.

In the factory:

- the temperature is measured by 10 sensors, each at a different location
- each sensor has a unique ID (1 to 10).

The procedure `CheckSensor()` will compare the measured temperature against each of two constant values, `LowTemp` and `HighTemp`. It will perform the following actions depending on the result of the comparison.

Measured temperature	Action
below `LowTemp`	Output `"Cold"`
from `LowTemp` to `HighTemp`	Output `"Normal"`
above `HighTemp`	Call procedure `Alarm()`

A library function, `GetTemp()`, returns the temperature value from a given sensor.

The structured English representing the algorithm for the procedure `CheckSensor()` is as follows:

1. Prompt for the input of a sensor ID.
2. Input a sensor ID.
3. If the sensor ID is invalid, repeat from step 1.
4. Call the `GetTemp()` function with the sensor ID as the parameter, to obtain the relevant temperature.
5. Compare the temperature against the two constant values and take the appropriate action.

Draw a program flowchart on the next page to represent the algorithm for procedure `CheckSensor()`.

Variable declarations are not required in program flowcharts.

Q3

3 **(a)** A student is developing an algorithm to search through a 1D array of 100 elements. Each element of the array, `Result`, contains a `REAL` value.

The algorithm will output:

- the average value of all the elements
- the number of elements with a value of zero.

The structured English description of the algorithm is:

1. SET Total value to 0
2. SET Zero count to 0
3. SELECT the first element
4. ADD value of element to Total value
5. IF element value is 0 then INCREMENT Zero count
6. REPEAT from step 4 for next element, until element is last element
7. SET Average to Total / 100
8. OUTPUT a suitable message and Average
9. OUTPUT a suitable message and Zero count

Write **pseudocode** for this algorithm.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [7]

(b) The student decides to change the algorithm and implement it as a procedure, `ScanArray()`, which will be called with three parameters.

```
ScanArray(AverageValue, ZeroCount, ArrayName)
```

`ScanArray()` will modify the first two parameters so that the new values are available to the calling program or module.

Write the **pseudocode** procedure header for `ScanArray()`.

.....

.....

.....

..... [4]

Q4

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

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

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

`"Bill Smith*081234567"`

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

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

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

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

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

Module	Description

`ProcessArray()`	• Check each element of the array:
	  ○ Read the student name from the array
	  ○ Ignore unused elements
	  ○ Call `SearchFile()` with the student name
	  ○ If the student name is found, call `AddToFile()` to write the student details to the class file
	  ○ If the student name is not found, call `AddToFile()` to write a new string to the class file, formed as follows:
	  <Name>"*No number"
	• Return the number of students who have not provided a telephone number

`SearchFile()`	• Search for a given student name at the start of each line in the file `StudentContact.txt`:
	  ○ If the search string is found, return the text line from `StudentContact.txt`
	  ○ If the search string is not found, return an empty string

`AddToFile()`	• Append the given string to a specified file, for example, `AddToFile(StringName, FileName)`

(a) Write **program code** for the module `SearchFile()`.

Visual Basic and Pascal: You should include the declaration statements for variables.
Python: You should show a comment statement for each variable used with its data type.

Programming language .....

Program code

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [8]

(b) Write **pseudocode** for the module `ProcessArray()`.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [9]

(c) `ProcessArray()` is modified to make it general purpose. It will now be called with two parameters as follows:

- an array
- a string representing the name of a class contact file

It will still return the number of students who have not provided a contact telephone number.

Write **program code** for the header (declaration) of the modified `ProcessArray()`.

Programming language .....

Program code

.....

.....

.....

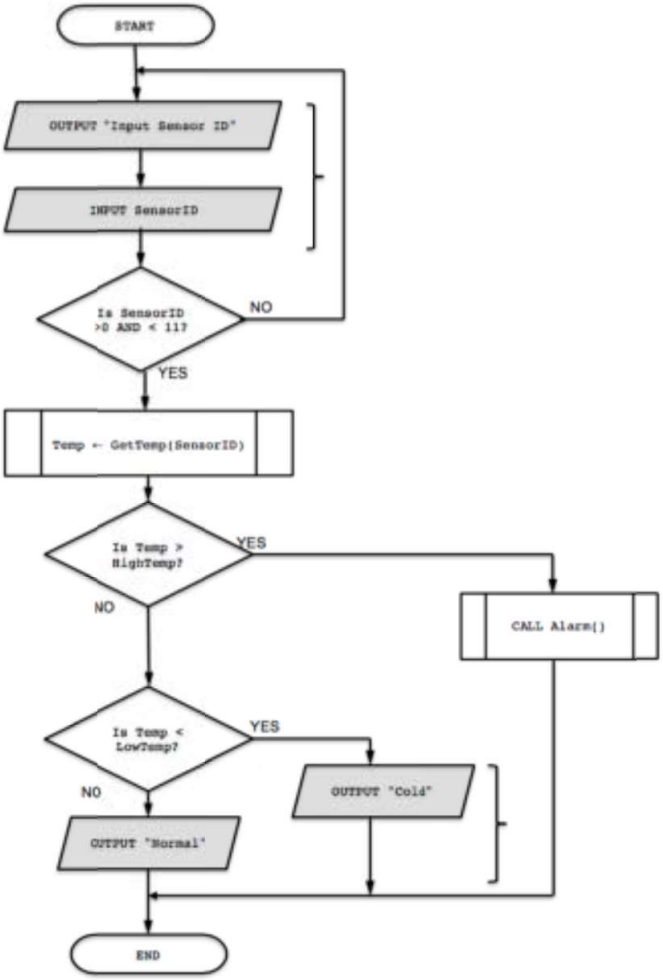
..... [3]

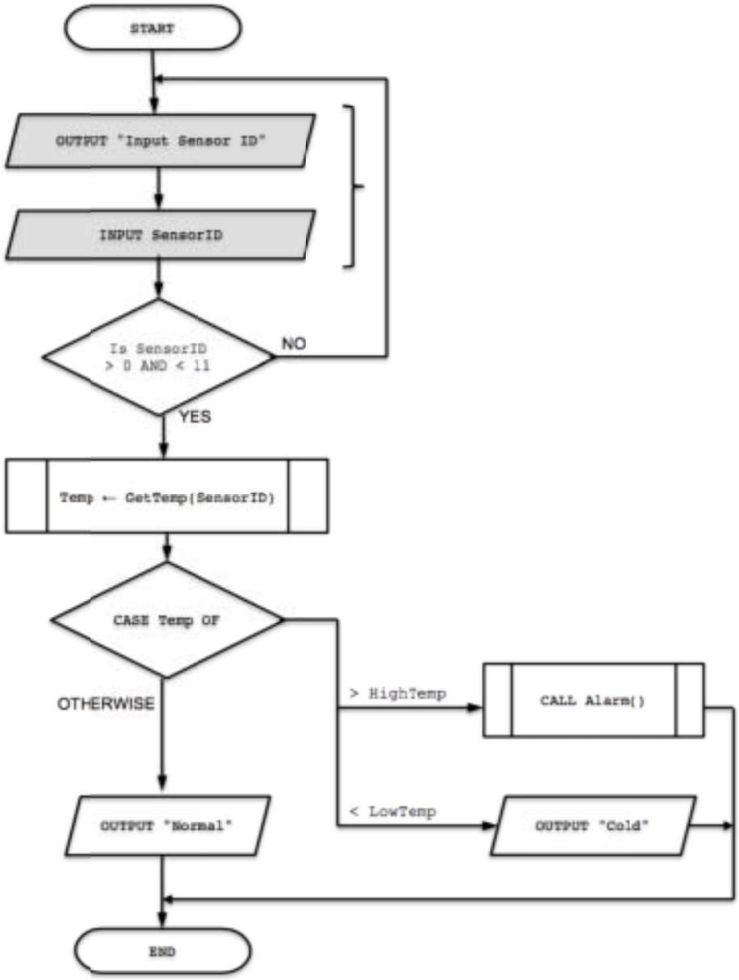
Mark Scheme

Q1

Question	Answer	Marks
2(a)(i)	• To make a more manageable / understandable solution • To support modular design	1
2(a)(ii)	• Allows the subroutine to be called from many / multiple places • Subroutine may be (independently) tested and debugged • If the task changes the change needs to be made only once • Reduces unnecessary duplication / program lines • Allows teams to work on different parts of the solution	3
2(a)(iii)	**Type of subroutine**: Function **Justification**: It returns a value // assigns a value to variable `Answer` One mark for type One mark for justification	2
3(b)	Key points: • to increase the level of detail of the algorithm // break the problem into smaller steps • until steps are easier to solve // to be directly translated into lines of code	2

Q2

Question	Answer	Marks
3	 One mark for: 1 START and END / STOP 2 prompt and input of `SensorID` (allow alt name) 3 decision box checking that `SensorID` is between 1 & 10 4 calling `GetTemp` with `SensorID` as parameter 5 decision box comparing `Temp` > `HighTemp` 6 calling `Alarm()` 7 decision box comparing `Temp` < `LowTemp` 8 both output messages	**8**

Question	Answer	Marks
3	ALTERNATIVE SOLUTION USING 'CASE'  Mark as follows: 1 One mark for START and END / STOP 2 One mark for prompt and input of `SensorID` (allow alt name) 3 One mark for decision box checking that `SensorID` is between 1 & 10 4 One mark for calling `GetTemp` with `SensorID` as parameter 5 One mark for decision box `CASE Temp` 6 One mark for calling `Alarm()` 7 One mark for two correct `CASE` conditions 8 One mark for both output messages	8

Q3

Question	Answer	Marks

3(a)	`TotalValue ← 0`
	`ZeroCount ← 0`
	
	`FOR Index ← 1 TO 100`
	`  TotalValue ← TotalValue + Result[Index]`
	`  IF Result[Index] = 0.0`
	`  THEN`
	`  ZeroCount ← ZeroCount + 1`
	`  ENDIF`
	`ENDFOR`
	
	`OUTPUT "The average is ", (TotalValue / 100)`
	`OUTPUT "The number of elements with a zero value is ",`
	`  ZeroCount`
	
	One mark for each of the following:
	
	1 Both initialisations
	2 Loop 100 times
	3 Adding individual element to TotalValue **in a loop**
	4 Check if element value is zero **in a loop**
	5 If so increment ZeroCount **in a loop**
	6 Average is calculated after the loop
	7 Both OUTPUT statements, including message and variables
		7

3(b)	`PROCEDURE ScanArray (BYREF AverageValue: REAL, _`
	`BYREF ZeroCount: INTEGER, ArrayName : ARRAY)`
	
	One mark for each underlined part
	
	Names unimportant but first two parameters must be `BYREF`
		4

Question	Answer	Marks

6(a)	'Pseudocode' solution included here for development and clarification of mark scheme.
	
	Programming language example solutions appear in the Appendix.
	
	`FUNCTION SearchFile (SearchString : STRING) RETURNS STRING`
	
	`  DECLARE FileData : STRING`
	`  DECLARE Found : BOOLEAN`
	`  DECLARE SearchLength : INTEGER`
	
	`  Found ← FALSE`
	`  SearchLength ← LENGTH(SearchString)`
	
	`  OPENFILE "StudentContact.txt" FOR READ`
	
	`  WHILE NOT EOF("StudentContact.txt") AND NOT Found`
	`  READFILE "StudentContact.txt", FileData`
	`  IF SearchString = LEFT(FileData, SearchLength)`
	`  THEN`
	`  Found ← TRUE`
	`  ENDIF`
	`  ENDWHILE`
	
	`  CLOSEFILE "StudentContact.txt"`
	
	`  IF NOT FOUND`
	`  THEN`
	`  RETURN ""`
	`  ELSE`
	`  RETURN FileData`
	`  ENDIF`
	
	`ENDFUNCTION`
	
	One mark for each of the following:
	
	1. Function header and end (where appropriate). Parameter optional but if present must be of type `STRING`
	2. Calculate length of string from parameter // extract substring from file line
	3. File `OPEN()` in `READ` mode **and** subsequent `CLOSE()`
	4. `WHILE` loop repeating until `EOF()`
	5. read a line from the file **in a loop**
	6. compare name from file with `SearchString` **in a loop**
	7. exit loop if `SearchString` found
	8. Return the line from the file if `SearchString` found **or** an empty string if not found
		8

Question	Answer	Marks

6(b)	
	`FUNCTION ProcessArray() RETURNS INTEGER`
	
	`  DECLARE NoTelNumber : INTEGER`
	`  DECLARE Index : INTEGER`
	`  DECLARE ThisName : STRING`
	`  DECLARE StudentData : STRING`
	
	`  NoTelNumber ← 0`
	
	`  FOR Index ← 1 to 40`
	`  ThisName ← ClassList[Index]`
	`  IF ThisName <> "" //Skip blanks`
	`  THEN`
	`  StudentData ← SearchFile(ThisName)`
	`  IF StudentData = "" //Student not found`
	`  THEN`
	`  StudentData ← ThisName & "*No number"`
	`  NoTelNumber ← NoTelNumber + 1`
	`  ENDIF`
	`  CALL AddToFile(StudentData, "ClassContact.txt")`
	`  ENDIF`
	
	`  ENDFOR`
	
	`  RETURN NoTelNumber`
	
	`ENDFUNCTION`
	
	One mark for each of the following:
	
	1. Function header and end, including return parameter
	2. Declaration and initialisation of local count variable (`NoTelNumber`)
	3. `FOR` loop for 40 array elements …
	4. skip empty elements **in a loop**
	5. use `SearchFile(ThisName)` and save return value **in a loop**
	6. if `Searchfile()` returns an empty string, add "*No number" to `SearchString` …
	7. ... and increment count
	8. call `AddToFile` with both parameters as above **in a loop**
	9. Return count **outside the loop**
		9

6(c)	'Pseudocode' solution included here for development and clarification of mark scheme.
	
	Programming language example solutions appear in the Appendix.
	
	`FUNCTION ProcessArray (ClassList : ARRAY, ClassContact : STRING)`
	`RETURNS : INTEGER`
	
	One mark per underlined section.
		3