8 Programming [15 MARKS]
ANSWERS

Question	Answer	Marks
10	• AO2 (maximum 9 marks) • AO3 (maximum 6 marks) **Data Structures required** names shown underlined must be used as given in the scenario 2D Array or list `Evening[1:10, 1:20] / Evening[0:9, 0:19]` Variables `Counter, SeatCounter, NumSeats, Row, Column` **Requirements (techniques)** **R1** Find number of seats available for each performance and output (searching, nested iteration, output) **R2** Inputs and validates number of seats (input, iteration, and selection) **R3** Checking if seats available (selection, assignment, output with appropriate messages) ***Example 15-mark answer in pseudocode***	**15**

```
// meaningful identifier names and appropriate data structures to store the data required
DECLARE Counter, SeatCounter, NumSeats, Row, Column : INTEGER

CONSTANT HouseFull = 200
CONSTANT MaxRow = 10
CONSTANT MaxColumn = 20

SeatCounter1 ← 0 // initialise seat counter for performance 1
```

Question	Answer	Marks
10		

```
FOR Row ← 1 TO 10
    FOR Column ← 1 TO 20
        IF Evening[Row, Column]
          THEN
            SeatCounter ← SeatCounter + 1
        ENDIF
    NEXT Column
NEXT Row

// validate input
OUTPUT "How many seats do you want to book? 1, 2, 3 or 4 "
INPUT NumSeats

WHILE 1 < NumSeats OR NumSeats > 4 OR NumSeats <> ROUND(NumSeats, 0)
    OUTPUT "Please enter 1, 2, 3 or 4 for the number of seats "
    INPUT NumSeats
ENDWHILE

IF SeatCounter + NumSeats > 200)// check for house full
  THEN
    OUTPUT "House full"
  ELSE
    IF SeatCounter + NumSeats > 200 // checks for not enough seats
            THEN
              OUTPUT "Only ", SeatCounter + NumSeats - 200, " seats left"
            ELSE
              FOR Counter ← 1 TO NumSeats // book required number of seats for performance
                  Evening[MOD(SeatCounter + Counter, MaxColumn), DIV(SeatCounter +
                          Counter), MaxColumn] ← TRUE
                  OUTPUT "Row ", MOD(SeatCounter + Counter, MaxColumn), " seat ",
                          DIV(SeatCounter + Counter, MaxColumn)," booked"
              NEXT Counter
           ENDIF
```

8 Programming [15 MARKS]
ANSWERS

Marking Instructions in italics			
AO2: Apply knowledge and understanding of the principles and concepts of computer science to a given context, including the analysis and design of computational or programming problems			
0	**1-3**	**4-6**	**7-9**
No creditable response.	At least one programming technique has been used. *Any use of selection, iteration, counting, totalling, input and output.*	Some programming techniques used are appropriate to the problem. *More than one technique seen applied to the scenario, check the list of techniques needed.*	The range of programming techniques used is appropriate to the problem. *All criteria stated for the scenario have been covered by the use of appropriate programming techniques, check list of techniques needed.*
	Some data has been stored but not appropriately. *Any **use** of variables or arrays or other language dependent data structures e.g. Python lists.*	Some of the data structures chosen are appropriate and store some of the data required. *More than one data structure **used** to store data required by the scenario.*	The data structures chosen are appropriate and store all the data required. *The data structures **used** store all the data required by the scenario.*

Marking Instructions in italics			
AO3: Provide solutions to problems by:			
	evaluating computer systems	**making reasoned judgements**	**presenting conclusions**
0	**1-2**	**3-4**	**5-6**
No creditable response.	Program seen without relevant comments.	Program seen with some relevant comment(s).	The program has been fully commented
	Some identifier names used are appropriate *Some of the data structures used have meaningful names.*	The majority of identifiers used are appropriately named. *Most of the data structures used have meaningful names.*	Suitable identifiers with names meaningful to their purpose have been used throughout. *All of the data structures used have meaningful names.*
	The solution is illogical.	The solution contains parts that may be illogical.	The program is in a logical order.
	The solution is inaccurate in many places. *Solution contains few lines of code with errors that attempt to perform a task given in the scenario.*	The solution contains parts that are inaccurate. *Solution contains lines of code with some errors that logically perform tasks given in the scenario. Ignore minor syntax errors.*	The solution is accurate. *Solution logically performs all the tasks given in the scenario. Ignore minor syntax errors.*
	The solution attempts at least one of the requirements. *Solution contains lines of code that attempt at least one task given in the scenario.*	The solution attempts to meet most of the requirements. *Solution contains lines of code that perform most tasks given in the scenario.*	The solution meets all the requirements given in the question. *Solution performs all the tasks given in the scenario.*

8 Programming [15 MARKS]
ANSWERS

Question	Answer	Marks
11	Requirements may be met using a suitable built-in function from the programming language used (Python, VB.NET or Java) Tables for AO2 and AO3 are used to award a mark in a suitable band using a best fit approach. Marks are available for: • AO2 (maximum 9 marks) • AO3 (maximum 6 marks) **Data Structures required** with names as given in the scenario The names underlined must be used as they are provided in the scenario: Arrays or lists `WoodType[]`, `Price[]`, `Customers[]`, `Quotations[]` **Requirements (techniques)** **R1** Input and store customer name, room length and width, with validation of input for room dimensions, including error message and repeated input (Input with prompts, range check and iteration). **R2** Initialise wood arrays. Calculate room area, select and store wood required. Determine cost of wood type and calculate price of wood to purchase. Round and store all data to relevant array (array initialisation, rounding, data retrieval from array, calculation and storage of results). **R3** Output full details: name of customer, choice of wood and quotation price with appropriate messages. Program continues for next customer (Output with messages, iteration of whole program).	**15**

8 Programming [15 MARKS]
ANSWERS

Question	Answer	Marks

11	**Example 15-mark answer in pseudocode**
	
	`// declarations not required in the answer`
	`// initial population of WoodType[] and Price[] arrays`
	`// input and loops are also acceptable`
	`WoodType[1] ← "Laminate"`
	`WoodType[2] ← "Pine"`
	`WoodType[3] ← "Oak"`
	`Price[1] ← 29.99`
	`Price[2] ← 39.99`
	`Price[3] ← 54.99`
	`// initialises starting customer in sales arrays`
	`CurrentCustomer ← 1`
	`// to allow program to continue to next customer`
	`Cont ← TRUE`
	`WHILE Cont DO`
	`// input customer name`
	`  OUTPUT "Input the customer's name "`
	`  INPUT Customers[CurrentCustomer]`
	`// input of room dimensions with validation`
	`  OUTPUT "What is the length of your room? "`
	`  INPUT RoomLength`
	`// validate RoomLength`
	`  WHILE RoomLength < 1.5 OR RoomLength > 10.0`
	`    OUTPUT "The measurement must be in the range 1.5 to 10.0 inclusive, please try again "`
	`    INPUT RoomLength`
	`  ENDWHILE`
	`  OUTPUT "What is the width of your room? "`
	`  INPUT RoomWidth`
	`// validate RoomWidth`
	
	`  WHILE RoomWidth < 1.5 OR RoomWidth > 10.0`
	`    OUTPUT "The measurement must be in the range 1.5 to 10.0 inclusive, please try again "`
	`    INPUT RoomWidth`
	`  ENDWHILE`
	`  RoomArea ← ROUND(RoomLength, 1) * ROUND(RoomWidth, 1)`
	`  RoomArea ← ROUND (RoomArea + 0.5, 0)`
	`// show the wood available and prices`
	`  OUTPUT "the wood choices available are:"`
	`  OUTPUT "Number  Wood Type  Price($)"`
	`  FOR Count ← 1 TO 3`
	`    OUTPUT Count, "  ", WoodType[Count], "  ",`	

8 Programming [15 MARKS]
ANSWERS

Question	Answer	Marks
11		

```
Price[Count]
    Next Count
// input wood choice
    OUTPUT "Input a number from 1 to 3 "
    INPUT WoodChoice
// validate wood choice
    WHILE WoodChoice < 1 OR WoodChoice > 3
        OUTPUT "Your input is out of range, please try
                again "
        INPUT WoodChoice
    ENDWHILE
// to calculate the total cost of the wood
    WoodCost ← RoomArea * Price[WoodChoice]
// to store the relevant data in Quotations[]
    Quotations[CurrentCustomer, 1] ← RoomLength
    Quotations[CurrentCustomer, 2] ← RoomWidth
    Quotations[CurrentCustomer, 3] ← RoomArea
    Quotations[CurrentCustomer, 4] ← WoodChoice
    Quotations[CurrentCustomer, 5] ← WoodCost

// final output of quotation
    OUTPUT "Customer name: ", Customers[CurrentCustomer]
    OUTPUT "The wood you have chosen is: ",
            WoodType[WoodChoice]
    OUTPUT "Your total price is: ",
           Quotations[CurrentCustomer,5]
// ready for next customer
    CurrentCustomer ← CurrentCustomer + 1
// resets CurrentCustomer to beginning of array when
   array
// limit reached
    IF CurrentCustomer > 100
        THEN
          CurrentCustomer ← 1
    ENDIF
ENDWHILE
```

8 Programming [15 MARKS]
ANSWERS

Question	Answer	Marks

10	• AO2 (maximum 9 marks)
	• AO3 (maximum 6 marks)
	
	Data Structures required names shown underlined must be used as given in the scenario
	2D Array or list Temperatures
	Variables MaxDay, MinDay, AvDay, MaxWeek, MinWeek, AvWeek
	
	Requirements (techniques)
	
	R1 Find maximum and minimum temperatures for each day and calculates the average daily temperature (searching, totalling)
	R2 Find maximum and minimum temperatures for week and calculates the average weekly temperature (nested searching, totalling)
	R3 outputs for each day name, the rounded values for maximum temperature, minimum temperatures and average temperature. Outputs for the week the rounded values for maximum temperature, minimum temperatures and average temperature (output with appropriate messages and rounded values)
	
	Example 15-mark answer in pseudocode:
	
	`// meaningful identifier names and appropriate data structures to store the data required`
	`DECLARE DayCounter, HourCounter : INTEGER`
	`DECLARE AvDay, AvWeek, MaxDay, MinDay, MaxWeek, MinWeek : REAL`
	`DECLARE DayTotal, WeekTotal : REAL`
	`DECLARE Day : STRING`
	
	`CONSTANT Hours ← 24`
	`CONSTANT Days ← 7`
		15

Question	Answer	Marks

10	`MaxWeek ← -1000// initialise max and min temperatures and total for the week`
	`MinWeek ← 1000`
	`WeekTotal ← 0`
	
	`FOR DayCounter ← 0 TO Days - 1`
	`  MaxDay ← -1000// initialise max and min temperatures and total for each day`
	`  MinDay ← 1000`
	`  DayTotal ← 0`
	`  FOR HourCounter ← 0 TO Hours - 1`
	`  DayTotal ← DayTotal + Temperatures(HourCounter, DayCounter)`
	`  // update total maximum and minimum`
	`  IF Temperatures(HourCounter, DayCounter) > MaxDay`
	`  THEN`
	`  MaxDay ← Temperatures(HourCounter, DayCounter)`
	`  ENDIF`
	`  IF Temperatures(HourCounter, DayCounter) < MinDay`
	`  THEN`
	`  MinDay ← Temperatures(HourCounter, DayCounter)`
	`  ENDIF`
	`  NEXT HourCounter`
	
	`  CASE OF DayCounter  // select message for day`
	`  0 : Day ← "Monday"`
	`  1 : Day ← "Tuesday"`
	`  2 : Day ← "Wednesday"`
	`  3 : Day ← "Thursday"`
	`  4 : Day ← "Friday"`
	`  5 : Day ← "Saturday"`
	`  6 : Day ← "Sunday"`
	
	`  ENDCASE`
	
	`  DayAverage ← DayTotal / Hours  // output results for day`
	`  OUTPUT Day  // Results from a day`
	`  OUTPUT "Maximum temperature ", MaxDay`
	`  OUTPUT "Minimum temperature ", MinDay`
	`  OUTPUT "Average temperature ", ROUND(DayAverage,2)`	

8 Programming [15 MARKS]
ANSWERS

Question	Answer	Marks

10	`IF MaxDay > MaxWeek  // update total maximum and minimum`
	`  THEN`
	`  MaxWeek ← MaxDay`
	`ENDIF`
	
	`IF MinDay > MinWeek`
	`  THEN`
	`  MinWeek ← MinDay`
	`ENDIF`
	`WeekTotal ← WeekTotal + DayTotal  // update total for week`
	
	`NEXT DayCounter`
	`WeekAverage ← WeekTotal / Days`
	
	`OUTPUT "Maximum temperature for week ", MaxWeek// output results for week`
	`OUTPUT "Minimum temperature for week ", MinWeek`
	`OUTPUT "Average temperature for Week ", ROUND(WeekAverage,2)`	

Question	Answer	Marks
11	Read the whole answer: Check if each requirement listed below has been met. Requirements may be met using a suitable built-in function from the programming language used (Python, VB.NET or Java). On place a SEEN mark if requirement met, cross if no attempt seen, omission mark and/or comment if partially met (see marked scripts). Use the tables for AO2 and AO3 below to award a mark in a suitable band using a best fit approach, then add up the total: • AO2 (maximum 9 marks) • AO3 (maximum 6 marks) **Data structures required:** The names underlined must match those given in the scenario: Arrays or lists `Days[]`, `Readings[]`, `AverageTemp[]` Variables `WeekLoop`, `DayLoop`, `InTemp`, `TotalDayTemp`, `TotalWeekTemp`, `AverageWeekTemp` **Requirements (techniques):** **R1** Input and store hourly temperatures and validation of input temperatures for each day (with prompts, range check and (nested)iteration) **R2** Calculate, round to one decimal place and store daily average temperatures and calculate the weekly average temperature rounded to one decimal place (iteration, totalling and rounding) **R3** Convert all average temperatures to Fahrenheit (to one decimal place) and output the average temperatures in both Celsius and Fahrenheit. Output with appropriate messages. (output and rounding)	**15**

Question	Answer	Marks

11	**Example 15 mark answer in pseudocode**
	
	`// meaningful identifiers and appropriate data structures for`
	`// all data required`
	`DECLARE Days : ARRAY[1:7] OF STRING`
	`DECLARE Readings : ARRAY[1:7, 1:24] OF REAL`
	`DECLARE AverageTemp : ARRAY[1:7] OF REAL`
	`DECLARE WeekLoop : INTEGER`
	`DECLARE DayLoop : INTEGER`
	`DECLARE InTemp : REAL`
	`DECLARE TotalDayTemp : REAL`
	`DECLARE TotalWeekTemp : REAL`
	`DECLARE AverageWeekTemp : REAL`
	`// initial population of Days[] array`
	`// input and a loop are also acceptable`
	`Days[1] ← "Sunday"`
	`Days[2] ← "Monday"`
	`Days[3] ← "Tuesday"`
	`Days[4] ← "Wednesday"`
	`Days[5] ← "Thursday"`
	`Days[6] ← "Friday"`
	`Days[7] ← "Saturday"`
	`// input temperatures inside nested loop`
	`FOR WeekLoop ← 1 TO 7`
	`  TotalDayTemp ← 0`
	`  FOR DayLoop ← 1 TO 24`
	`  OUTPUT "Enter temperature ", DayLoop, " for ", Days[WeekLoop]`	

8 Programming [15 MARKS]

ANSWERS

Question	Answer	Marks

11	INPUT InTemp
	// validation of input for between -20 and +50 inclusive
	WHILE InTemp < -20.0 OR InTemp > 50.0 DO
	OUTPUT "Your temperature must be between -20.0 and +50.0 inclusive. Please try again"
	INPUT InTemp
	ENDWHILE
	Readings[WeekLoop, DayLoop] ← InTemp
	// totalling of temperatures during the day
	TotalDayTemp ← TotalDayTemp + ROUND(InTemp, 1)
	NEXT DayLoop
	
	// average temperature for the day
	AverageTemp[WeekLoop] ← ROUND(TotalDayTemp / 24,1)
	NEXT WeekLoop
	// calculate the average temperature for the week
	TotalWeekTemp ← 0
	FOR WeekLoop ← 1 TO 7
	TotalWeekTemp ← TotalWeekTemp + AverageTemp[WeekLoop]
	NEXT WeekLoop
	
	AverageWeekTemp ← ROUND(TotalWeekTemp / 7,1)
	// outputs in Celsius and Fahrenheit
	FOR WeekLoop ← 1 TO 7
	OUTPUT "The average temperature on ", Days[WeekLoop], " was ", AverageTemp[WeekLoop], " Celsius and ", ROUND(AverageWeekTemp * 9 / 5 + 32), 1, " Fahrenheit"
	NEXT WeekLoop
	
	OUTPUT "The average temperature for the week was ", AverageWeekTemp," Celsius and ", ROUND(AverageWeekTemp * 9 / 5 + 32, 1),"Fahrenheit"	

Question	Answer	Marks

12	Read the whole answer:
	Check if each requirement listed below has been met. Requirements may be met using a suitable built-in function from the programming language used (Python, VB.NET or Java)
	Mark SEEN on script if requirement met, cross if no attempt seen, NE if partially met (see marked scripts).
	Use the tables for A02 and A03 below to award a mark in a suitable band using a best fit approach
	Then add up the total.
	Marks are available for:
	• AO2 (maximum 9 marks)
	• AO3 (maximum 6 marks)
	
	Data Structures required names shown underlined must be used as given in the scenario
	Arrays or lists `Account, AccDetails`
	Variable `Size, AccountNumber`
	
	Requirements (techniques)
	R1 Check account number and password (iteration and validation, selection, input, output)
	R2 Display menu and make a selection (output, input and selection)
	R3 Perform actions selected (use of arrays and procedures with parameters)
	
	Example 15 mark answer in pseudocode
	// Procedures to be called
	PROCEDURE CheckDetails(AccID : INTEGER)
	DECLARE Name, Password : STRING // local variables
	Valid ← FALSE
	IF AccID <0 OR AccID > Size
	THEN
	OUTPUT "Invalid Account Number"
	ELSE
	OUTPUT "Please Enter Name "
	INPUT Name
	OUTPUT "Please Enter Password "
	INPUT Password
	IF Name <> Account[AccID,1] OR Password <> Account[AccID,2]
	THEN
	OUTPUT "Invalid name or password"
	ELSE
		15

8 Programming [15 MARKS]
ANSWERS

12

```
            Valid ← True
        ENDIF
    ENDIF
ENDPROCEDURE

PROCEDURE Balance(AccID : INTEGER)
    OUTPUT "Your balance is ", AccDetails[AccID,1]
ENDPROCEDURE

PROCEDURE WithDrawal(AccID : INTEGER)
    DECLARE Amount : REAL // local variable
    REPEAT
        OUTPUT "Please enter amount to withdraw "
        INPUT Amount
        IF Amount > AccDetails[AccID,3]
          THEN
            OUTPUT "Amount greater than withdrawal limit"
        ENDIF
        IF Amount > AccDetails[AccID,2] + AccDetails[AccID,1]
          THEN
            OUTPUT "Amount greater than cash available"
        ENDIF
        IF Amount <= AccDetails[AccID,3] AND Amount < AccDetails[AccID,2] +
          AccDetails[AccID,1]
          THEN
            AccDetails[AccID,1] ← AccDetails[AccID,1] - Amount
        ENDIF
    UNTIL Amount<= AccDetails[AccID,3] AND Amount > AccDetails[AccID,2] +
          AccDetails[AccID,1] AND Amount > 0
ENDPROCEDURE

PROCEDURE Deposit(AccID : INTEGER)
    DECLARE Amount : REAL // local variable
    REPEAT
        OUTPUT "Please enter a positive amount to deposit "
        INPUT Amount
    UNTIL Amount >0
    AccDetails[AccID,1] ← AccDetails[AccID,1] + Amount
```

12

```
ENDPROCEDURE

// Declarations of global variables for information - not required in candidate responses
DECLARE AccountNumber, Choice : INTEGER
DECLARE Valid, Exit : BOOLEAN

OUTPUT "Please enter your account number "
INPUT AccountNumber
CheckDetails(AccountNumber)

IF Valid
  THEN
    REPEAT
        OUTPUT "Menu"
        OUTPUT "1. display balance"
        OUTPUT "2. withdraw money"
        OUTPUT "3. deposit money"
        OUTPUT "4. exit"
        OUTPUT "please choose 1, 2, 3 or 4"
        INPUT Choice
        CASE OF Choice
          1 : Balance(AccountNumber)
          2 : Withdrawal(AccountNumber)
          3 : Deposit(AccountNumber)
          4 : Exit ← TRUE
          OTHERWISE OUTPUT "Invalid choice"
        ENDCASE
    UNTIL Choice = 4
  ELSE
    OUTPUT "Invalid account number "
ENDIF
```

8 Programming [15 MARKS]
ANSWERS

Question	Answer	Marks
11	Read the whole answer: Check if each requirement listed below has been met. Requirements may be met using a suitable built-in function from the programming language used (Python, VB.NET or Java). Mark SEEN on script if requirement met, cross if no attempt seen, NE if partially met (see marked scripts). Use the tables for A02 and A03 below to award a mark in a suitable band using a best fit approach. Then add up the total. Marks are available for: • AO2 (maximum 9 marks) • AO3 (maximum 6 marks) **Data structures required:** The names underlined must match those given in the scenario: Arrays or lists `Contacts[]` Variables `CurrentSize`, `Cont`, `Choice`, `NewContacts`, `Count`, `Count2`, `Flag` **Requirements (techniques):** **R1** Output menu and input choice, with validation (range check, output with messages, input with prompts). **R2** Input number of new entries, within limits, update current size of contacts, input new data and sort the array (range check, totalling, iteration and bubble sort). **R3** Output array whole contents and delete contents of array (iteration, output with labelling/messages, array initialisation).	**15**

Question	Answer	Marks

11	**Example 15 mark answer in pseudocode**
	
	`// meaningful identifiers and appropriate data structures for`
	`// all data required`
	`DECLARE Contacts : ARRAY[1:100, 1:2] OF STRING`
	`DECLARE CurrentSize : INTEGER`
	`DECLARE Cont : BOOLEAN`
	`DECLARE Choice : INTEGER`
	`DECLARE NewContacts : INTEGER`
	`DECLARE Count : INTEGER`
	`DECLARE Count2 : INTEGER`
	`DECLARE Flag : BOOLEAN`
	`DECLARE Temp1 : STRING`
	`DECLARE Temp2 : STRING`
	
	`// the number of contacts in the array`
	`CurrentSize ← 0`
	
	`// to allow program to continue indefinitely`
	`Cont ← TRUE`
	`WHILE Cont DO`
	`// display menu`
	`  OUTPUT "Please choose one of the following: "`
	`  OUTPUT "Press 1 to enter new contacts "`
	`  OUTPUT "Press 2 to display your contacts "`
	`  OUTPUT "Press 3 to delete all contacts "`
	`  INPUT Choice`
	`// validate choice as 1, 2 or 3`
	`  WHILE Choice = 1 AND CurrentSize = 100 DO`
	`  OUTPUT "Your contacts are full, please enter 2 or 3"`
	`  INPUT Choice`
	`  ENDWHILE`
	`  WHILE Choice < 1 OR Choice > 3 DO`
	`  OUTPUT "Incorrect entry - please enter 1, 2, or 3"`
	`  INPUT Choice`
	`  ENDWHILE`	

8 Programming [15 MARKS]
ANSWERS

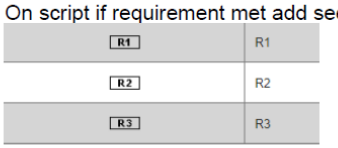
Question	Answer	Marks
11		

```
// enter new contacts
    IF Choice = 1
      THEN
        OUTPUT "How many contacts (1 to 5 only)?"
        INPUT NewContacts
// validates new contacts input
        WHILE NewContacts < 1 OR NewContacts > 5 DO
            OUTPUT "You may only enter between 1 and 5 contacts. Please try again"
            INPUT NewContacts
        ENDWHILE
// checks the maximum size is not exceeded
        WHILE CurrentSize + NewContacts > 100
            OUTPUT "Not enough space in your contacts"
            OUTPUT "The maximum number you may input is ", 100 - CurrentSize
            INPUT NewContacts
        ENDWHILE
        FOR Count ← CurrentSize + 1 TO CurrentSize + NewContacts
            OUTPUT "Enter the contact name as last name, first name"
            INPUT Contacts[Count, 1]
            OUTPUT "Enter the telephone number"
            INPUT Contacts[Count, 2]
        NEXT Count
        CurrentSize ← CurrentSize + NewContacts
// bubble sort to sort array if it contains 2 or more contacts
        IF CurrentSize >= 2
          THEN
            REPEAT
                Flag ← FALSE
                FOR Count ← 1 TO CurrentSize-1
                    IF Contacts[Count + 1, 1] <
                      Contacts[Count, 1]
                        THEN
                            Flag ← TRUE
                            Temp1 ← Contacts[Count, 1]
                            Temp2 ← Contacts[Count, 2]
                            Contacts[Count, 1] ← Contacts[Count + 1, 1]
                            Contacts[Count, 2] ← Contacts[Count + 1, 2]
                            Contacts[Count + 1, 1] ← Temp1
                            Contacts[Count + 1, 2] ← Temp2

                    ENDIF
                NEXT Count
            UNTIL NOT Flag
        ENDIF
    ENDIF
// display all contacts
    IF Choice = 2
      THEN
        IF CurrentSize > 0
          THEN
            OUTPUT "Name and Telephone Number"
            FOR Count ← 1 TO CurrentSize
                OUTPUT Contacts[Count, 1], "   ", Contacts[Count, 2]
                NEXT Count
        ENDIF
    ENDIF
// delete all contacts
    IF Choice = 3
      THEN
        FOR Count ← 1 TO 100
            FOR Count2 ← 1 TO 2
                Contacts[Count, Count2] ← ""
            NEXT Count2
        NEXT Count
    ENDIF
ENDWHILE
```

8 Programming [15 MARKS]

ANSWERS

Question	Answer	Marks
11	Read and understand the question before starting to mark any scripts. Read the whole answer before marking a script: Check if each requirement listed below has been met. Requirements may be met using a suitable built-in function from the programming language used (Python, VB.NET or Java) On script if requirement met add seen, NE if partial attempt, cross if no attempt (see marked scripts).  Use the tables for A02 and A03 below to award a mark in a suitable band using a best fit approach, then add up the total. Marks are available for: • AO2 (maximum 9 marks) • AO3 (maximum 6 marks)  **Data Structures required** with names as given in the scenario Arrays or lists `TeamName`, `TeamPoints` Variables `LeagueSize`, `MatchNo` **Requirements (techniques)** **R1** calculates total points for all matches played by each team (nested iteration, totalling) **R2** counts and outputs, with the team's name, for each team the total number of away wins, home wins, drawn matches and lost matches (nested iteration, counting, output) **R3** finds and outputs the name of the team with the highest number of points and the name of the team with the lowest number of points. (output, selection)	**15**

Question	Answer	Marks

11	***Example 15-mark answer in pseudocode:***
	`// meaningful identifier names and appropriate data structures to store the data required`
	`DECLARE TeamCounter : INTEGER`
	`DECLARE MatchCounter : INTEGER`
	`FOR TeamCounter ← 1 to LeagueSize // zero totals for each club's results`
	`  TotalPoints[TeamCounter] ← 0`
	`NEXT TeamCounter`
	
	`FOR TeamCounter ← 1 TO LeagueSize`
	`  AwayWinNo ← 0  // zero totals for each club's result details`
	`  HomeWinNo ← 0`
	`  DrawNo ← 0`
	`  LostNo ← 0`
	`  FOR MatchCounter ← 1 TO MatchNo`
	`  TotalPoints[TeamCounter] ← TotalPoints[TeamCounter] +`
	`  TeamPoints[TeamCounter, MatchCounter]`
	`  CASE OF TeamPoints[TeamCounter, MatchCounter]`
	`  3 : AwayWinNo ← AwayWinNo + 1`
	`  2 : HomeWinNo ← HomeWinNo + 1`
	`  1 : DrawNo ← DrawNo + 1`
	`  0 : LostNo ← LostNo + 1`
	`  ENDCASE`
	`  NEXT MatchCounter`
	
	`  OUTPUT "Team ", TeamName[TeamCounter]  // Output details of a team's results`
	`  OUTPUT "Total points ", TotalResult[TeamCounter]`
	`  OUTPUT "Away wins ", AwayWinNo`
	`  OUTPUT "Home wins ", HomeWinNo`
	`  OUTPUT "Draws ", DrawNo`
	`  OUTPUT "Losses ", LostNo`	

8 Programming [15 MARKS]
ANSWERS

Question	Answer	Marks

11	`// Check for highest and lowest results`
	`  IF TeamCounter = 1`
	`  THEN`
	`  HighestResult ← TotalPoints[TeamCounter]`
	`  LowestResult ← TotalPoints[TeamCounter]`
	`  ENDIF`
	
	`  IF TotalPoints[TeamCounter] > HighestResult`
	`  THEN`
	`  HighestResult ← TotalPoints[TeamCounter]`
	`  TopTeam ← TeamCounter`
	`  ENDIF`
	`  IF TotalPoints[TeamCounter] < LowestResult`
	`  THEN`
	`  LowestResult ← TotalPoints[TeamCounter]`
	`  BottomTeam ← TeamCounter`
	`  ENDIF`
	
	`NEXT TeamCounter`
	
	`// output names of the teams with the highest and lowest number of points`
	`OUTPUT "Top Team ", TeamName[TopTeam]`
	`OUTPUT "Bottom Team ", TeamName[BottomTeam]`	