8. Programming

8.2 Arrays

1 Arrays are data structures used in programming. Explain what is meant by the terms dimension and index in an array. Use examples of arrays in your explanations.

Dimension .....

.....

.....

.....

.....

Index .....

.....

.....

.....

.....

[3]

2 A one-dimensional array `dataArray[1:20]` needs each element set to zero.

(a) Write a pseudocode routine that sets each element to zero. Use the most suitable loop structure.

.....

.....

.....

.....

.....

.....

.....

..... [3]

(b) Explain why you chose this loop structure.

.....

..... [1]

3 **(a)** Describe a one-dimensional array. Include an example of an array declaration.

.....

.....

.....

.....

.....

..... [3]

(b) Explain how indexing could be used to search for a value stored in a one-dimensional array.

.....

.....

.....

..... [2]

4 Identify **three** different loop structures that you can use when writing pseudocode.

1 .....

.....

2 .....

.....

3 .....

.....[3]

5 **(a)** Write an algorithm, using pseudocode and a `FOR ... TO ... NEXT` loop structure, to input 1000 numbers into an array.

.....

.....

.....

.....

.....

.....[2]

(b) Rewrite your algorithm using another loop structure.

.....

.....

.....

.....

.....

.....

.....[4]

6 `REPEAT ... UNTIL` is one type of loop structure.

Identify and describe **two** other types of loop structure that you could use when writing pseudocode.

Loop structure 1.....

Description.....

.....

Loop structure 2.....

Description.....

.....[4]

7 `REPEAT ... UNTIL` and `WHILE ... DO ... ENDWHILE` are two different loop structures you can use when writing pseudocode.

Explain, using examples, why you would choose to use each type of loop.

Example 1 .....

.....

.....

.....

Reason for choice .....

.....

.....

Example 2 .....

.....

.....

.....

Reason for choice .....

.....

.....[6]

8 **(a)** Describe the purpose of each statement in this algorithm.

```
FOR I ← 1 TO 300
  INPUT Name[I]
NEXT I
```

.....

.....

.....

.....

.....[2]

(b) Identify, using pseudocode, another loop structure that the algorithm in **part (a)** could have used.

.....

.....[1]

(c) Write an algorithm, using pseudocode, to input a number between 0 and 100 inclusive. The algorithm should prompt for the input and output an error message if the number is outside this range.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[3]

9 **(a)** **Four** pseudocode descriptions and **five** pseudocode statements are shown. Draw one line to link each pseudocode description to the correct pseudocode statement. Not all pseudocode statements will be used.

Pseudocode description	Pseudocode statement
A loop that will iterate at least once.	FOR...TO...NEXT
A conditional statement to deal with many possible outcomes.	IF...THEN...ELSE...ENDIF
A loop that will iterate a set number of times.	WHILE...DO...ENDWHILE
A conditional statement with different outcomes for true and false.	CASE...OF...OTHERWISE...ENDCASE
	REPEAT...UNTIL

[4]

(b) Write an algorithm in pseudocode, using a single loop, to print 50 names that have been stored in an array.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[3]

10 Identify and describe **three** loop structures that are available in pseudocode.

Loop structure 1 .....

.....

Description .....

.....

.....

Loop structure 2 .....

.....

Description .....

.....

.....

Loop structure 3 .....

.....

Description .....

.....

.....

[6]

11 Name the **three** types of loop structure used in pseudocode.

.....

.....

.....

.....

.....

..... [3]

12 This pseudocode algorithm allows 5000 numbers to be entered and stored in an array called `Number`.

```
FOR Count ← 1 TO 5000
  INPUT Number[Count]
NEXT Count
```

Extend and re-write the algorithm using pseudocode to also count and output how many of the numbers stored in the array are greater than 500, using the variable `Higher`. Only output `Higher` once with an appropriate message.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [6]

13 **Four** pseudocode statements and **five** pseudocode uses are shown.

(a) Draw **one** line to link each pseudocode statement to the most appropriate pseudocode use.

Not all pseudocode uses will be required.

Pseudocode statement	Pseudocode use
`CALL Colour(NewColour)`	counting
`Value ← (A1 + A2 + A3) / 3`	finding an average
`Loop1 ← Loop1 + 1`	totalling
`IF Count > 7 THEN X1 ← 0`	using a conditional statement
	using a procedure

[4]

(b) A one-dimensional (1D) array called `Temperatures[]` has 25 elements beginning at index 1. It holds values that range between –20 and 100 inclusive.

Write a pseudocode algorithm using a single loop to find the lowest value in this array and output the result only once.

You do **not** need to declare or populate this array.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [4]

14 The string operation `SUBSTRING(Quote, Start, Number)` returns a string from `Quote` beginning at position `Start` that is `Number` characters long. The first character in `Quote` is in position 1.

Write pseudocode statements to:

- store the string `"Learning Never Exhausts The Mind"` in `Quote`
- extract and display the words `"The Mind"` from the string
- output the original string in lower case.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [5]

Pseudocode, Program code and Arrays

1 A one-dimensional (1D) array `Days[]` contains the names of the days of the week. A two-dimensional (2D) array `Readings[]` is used to store 24 temperature readings, taken once an hour, for each of the seven days of the week. A 1D array `AverageTemp[]` is used to store the average temperature for each day of the week.

The position of any day's data is the same in all three arrays. For example, if Wednesday is in index 4 of `Days[]`, Wednesday's temperature readings are in index 4 of `Readings[]` and Wednesday's average temperature is in index 4 of `AverageTemp[]`

The temperature readings are in Celsius to one decimal place. Temperatures can only be from –20.0 °C to +50.0 °C inclusive.

Write a program that meets the following requirements:

- input and validate the hourly temperatures for one week
- calculate and store the average temperature for each day of the week
- calculate the average temperature for the whole week
- convert all the average temperatures from Celsius to Fahrenheit by using the formula Fahrenheit = Celsius * 9/5 + 32
- output the average temperature in Celsius and in Fahrenheit for each day
- output the overall average temperature in Celsius and in Fahrenheit for the whole week.

You must use pseudocode or program code **and** add comments to explain how your code works.

You do **not** need to declare any arrays, variables or constants; you may assume that this has already been done.

All inputs and outputs must contain suitable messages.

All data output must be rounded to one decimal place.

You will need to initialise and populate the array `Days[]` at the start of the program.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [15]

2 A two-dimensional (2D) array `Account[]` contains account holders' names and passwords for a banking program.

A 2D array `AccDetails[]` has three columns containing the following details:

- column one stores the balance – the amount of money in the account, for example 250.00
- column two stores the overdraft limit – the maximum total amount an account holder can borrow from the bank after the account balance reaches 0.00, for example 100.00
- column three stores the withdrawal limit – the amount of money that can be withdrawn at one time, for example 200.00

The amount of money in a bank account can be negative (overdrawn) but **not** by more than the overdraft limit.
For example, an account with an overdraft limit of 100.00 must have a balance that is greater than or equal to –100.00

Suitable error messages must be displayed if a withdrawal cannot take place, for example if the overdraft limit or the size of withdrawal is exceeded.

The bank account ID gives the index of each account holder's data held in the two arrays.
For example, account ID 20's details would be held in:
`Account[20,1]` and `Account[20,2]`
`AccDetails[20,1]` `AccDetails[20,2]` and `AccDetails[20,3]`

The variable `Size` contains the number of accounts.

The arrays and variable `Size` have already been set up and the data stored.

Write a program that meets the following requirements:

- checks the account ID exists and the name and password entered by the account holder match the name and password stored in `Account[]` before any action can take place
- displays a menu showing the four actions available for the account holder to choose from:
    1. display balance
    2. withdraw money
    3. deposit money
    4. exit
- allows an action to be chosen and completed. Each action is completed by a procedure with a parameter of the account ID.

You must use pseudocode or program code **and** add comments to explain how your code works. All inputs and outputs must contain suitable messages.

You only need to declare any local arrays and local variables that you use.

You do **not** need to declare and initialise the data in the global arrays `Account[]` and `AccDetails[]` and the variable `Size`

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [15]

3 A two-dimensional (2D) array `Contacts[]` is used to store names and telephone numbers. All the data is stored as strings. The array must have the capacity to store 100 contacts in the form of:

- column 1 – contact names as: last name, first name
  for example: Smith, John
- column 2 – telephone numbers.

The variable `CurrentSize` shows how many contacts are in the array.

Write a program that meets the following requirements:

- display a menu of choices:
  - enter new contact details
  - display all the contact details
  - delete all the contact details
- validate the menu input
- allow up to a maximum of five new contacts to be added to the array at any one time
- do **not** allow more than 100 contacts in total
- after new contacts have been added, sort the array by contact name, as long as there are at least two contacts in the array
- output the whole of the array
- delete the contents of the array.

You must use pseudocode or program code **and** add comments to explain how your code works.

You do **not** need to declare any arrays, variables or constants; you may assume that this has already been done.

All inputs and outputs must contain suitable messages.

You do **not** need to initialise the data in the array `Contacts[]` and the variable `CurrentSize`

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [15]

4 Drama students put on a performance of a play for one evening. Seats in a small theatre can be booked for this performance.

The theatre has 10 rows of 20 seats. The status of the seat bookings for the evening is held in the two-dimensional (2D) Boolean array `Evening[]`

Each element contains `FALSE` if the seat is available and `TRUE` if the seat is booked.

Up to and including four seats can be booked at one time. Seats are allocated in order from those available. A row or seat number cannot be requested.

The array `Evening[]` has already been set up and some data stored.

Write a program that meets the following requirements:

- counts and outputs the number of seats already booked for the evening
- allows the user to input the number of seats required
- validates the input
- checks if enough seats are available:
  - if they are available
    - changes the status of the seats
    - outputs the row number and seat number for each seat booked
  - if they are **not** available:
    - outputs a message giving the number of seats left or 'House full' if the theatre is fully booked.

You must use pseudocode or program code **and** add comments to explain how your code works.
You do **not** need to declare any arrays or variables; you may assume that this has already been done.
You do **not** need to initialise the data in the array `Evening[]`
All inputs and outputs must contain suitable messages.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [15]

5 A wood flooring company stores the names of up to 100 customers in a one-dimensional (1D) array `Customers[]`. A two-dimensional (2D) array `Quotations[]` stores details of each customer's quotation:

- length of room (one decimal place)
- width of room (one decimal place)
- area of wood required (rounded up to next whole number)
- choice of wood index (whole number)
- price of wood required in dollars (two decimal places).

The floor measurements (room length and room width) are taken in metres. All floors are rectangles and room measurements must be between 1.5 and 10.0 inclusive.

The index of any customer's data is the same in both arrays. For example, a customer named in index 4 of `Customers[]` corresponds to the data in index 4 of `Quotations[]`

The wood choices available are:

Index	Wood type	Price per square metre ($)
1	Laminate	29.99
2	Pine	39.99
3	Oak	54.99

The data are stored in two 1D arrays named `WoodType[]` and `Price[]`. The index of the wood type and price in their arrays share the same index number.

Write a program that meets the following requirements:

- input a new customer's name, room length and room width
- check that each measurement is valid
- output an error message and require the measurement to be re-entered until it is valid
- calculate the area of the room by multiplying together the length of the room and the width of the room
- input the choice of wood and find its price per square metre
- calculate the price of the wood needed
- store all data in the relevant array
- output the customer's quotation to include: the name of the customer, the choice of wood and the calculated price of the wood required
- continue to accept the next customer.

You must use pseudocode or program code **and** add comments to explain how your code works. You do **not** need to declare any arrays or variables; you may assume that this has already been done.

You will need to initialise `WoodType[]` and `Price[]`

All inputs and outputs must contain suitable messages.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [15]

6 A weather station takes temperature readings once an hour for a week. These temperatures are stored in a two-dimensional (2D) array `Temperatures[]`
Each column contains 24 readings for a single day. The first temperature is recorded at 00:00 and the final temperature at 23:00. There are seven columns, one for each day of the week, starting with Monday and ending with Sunday.

The variables `MaxDay`, `MinDay` and `AvDay` are used to store the maximum, minimum, and average temperatures for a day. The variables `MaxWeek`, `MinWeek` and `AvWeek` are used to store the maximum, minimum, and average temperatures for the week.

The array has already been set up and the data stored.

Write a program that meets the following requirements:

- finds the maximum and minimum temperatures for each day
- calculates the average temperature for each day
- outputs for each day:
  - name of the day, for example Monday
  - maximum temperature
  - minimum temperature
  - average temperature
- finds the maximum and minimum temperatures for the week
- calculates the average temperature for the week
- outputs:
  - maximum temperature for the week
  - minimum temperature for the week
  - average temperature for the week.

All temperatures output must be rounded to two decimal places.

You must use pseudocode or program code **and** add comments to explain how your code works. All inputs and outputs must contain suitable messages.

You do **not** need to declare any arrays or variables; you may assume that this has already been done.
You do **not** need to initialise the data in the array `Temperatures[]`

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [15]

7 The one-dimensional (1D) array `Clubs[]` is used to store the names of 12 cricket clubs in a local sports league.

The two-dimensional (2D) array `Statistics[]` is used to store, for each cricket club, the number of:

- matches won
- matches drawn
- matches lost.

The 1D array `Points[]` is used to store the total number of points each cricket club has been awarded.

The position of any cricket club's data is the same in all three arrays. For example, the data in index 2 of `Statistics[]` and index 2 of `Points[]` belongs to the cricket club in index 2 of `Clubs[]`

The variable `Matches` stores the number of matches played by each team. Each team plays the same number of matches.

Points are awarded for:

- a win – 12 points
- a draw – 5 points
- a loss – 0 points.

Write a program that meets the following requirements:

- allows the number of matches played to be input and stored, with a maximum of 22 matches
- validates the number of matches played
- allows the names of the cricket clubs to be input and stored
- allows the number of matches won, drawn and lost to be input and stored for each team
- validates the number of matches won, drawn or lost against the number of matches played
- asks the user to re-enter the number of matches won, drawn or lost if the total does not match the number of matches played
- calculates and stores the total number of points for each club
- finds the cricket club or clubs with the highest number of points
- outputs the name or names of the winning club or clubs, the number of wins and the total number of points awarded.

You must use pseudocode or program code **and** add comments to explain how your code works.

You do **not** need to declare any arrays or variables; you may assume that this has already been done.

All inputs and outputs must contain suitable messages.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

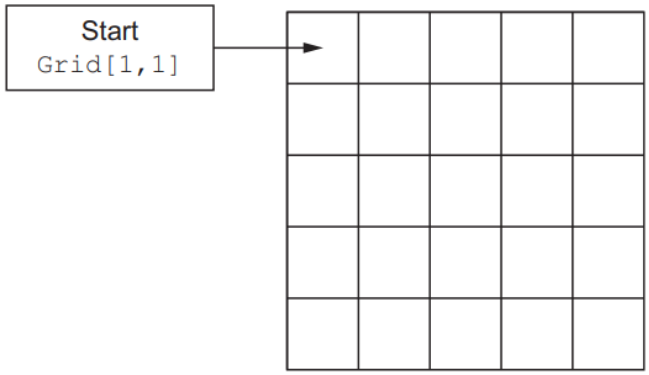
.....

.....

.....

..... [15]

8 A one-player game uses the two-dimensional (2D) array `Grid[]` to store the location of a secret cell to be found by the player in 10 moves. Each row and column has 5 cells.



At the start of the game:

- The program places an 'X' in a random cell (**not** in `Grid[1,1]`) and empties all the other cells in the grid.
- The player starts at the top left of the grid.
- The player has 10 moves.

During the game:

- The player can move left, right, up or down by one cell and the move must be within the grid.
- "You Win" is displayed if the player moves to the cell with 'X' and has played 10 moves or less.
- "You Lose" is displayed if the player has made 10 moves without finding the 'X'.

Write a program that meets these requirements.

You must use pseudocode or program code **and** add comments to explain how your code works.

You do **not** need to declare any arrays or variables; you may assume that this has already been done.

All inputs and outputs must contain suitable messages.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [15]

9 A one-dimensional (1D) array `Teams[]` contains the names of 10 football teams in a local league. A two-dimensional (2D) array `Results[]` stores, for each team, the total number of:

- games won
- games drawn
- games lost
- points.

The position of any team's data is the same in both arrays. For example the data in index 3 of `Results[]` belongs to the team in index 3 of `Teams[]`

The array data will be used to find the current leader of the league.

The variable `Played` stores the number of games played by each team. Each team plays the same number of games.

Write a program that meets the following requirements:

- allows the number of games played to be input and stored, with a maximum of 18 games
- allows the names of the teams to be input and stored
- allows the number of games won, drawn and lost to be input and stored for each team
- validates the number of games played and the number of games won, drawn or lost against the number of games played
- calculates and stores the number of points for each team using three points for a win and one point for a draw; there are no points for a loss
- sorts the array `Results[]` into descending order of number of points, ensuring the corresponding parallel array `Teams[]` is kept in the same order
- determines how many teams have the highest number of points
- outputs the name(s) of the winning team(s) along with the number of points achieved.

You must use pseudocode or program code **and** add comments to explain how your code works.

You do **not** need to declare any arrays, variables or constants; you may assume that this has already been done.

All inputs and outputs must contain suitable messages.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [15]

10 A running club has 200 members who compete in a 1-kilometre running competition every month.
Members' names are stored in the one-dimensional (1D) array `MemberName[]`
Each member's time, in seconds, for the 1-kilometre run will be stored in another one-dimensional (1D) array `MemberTime[]`
The position of each member's data in the two arrays is the same. For example, the member stored at index 10 in `MemberName[]` and at index 10 in `MemberTime[]` is the same.

The running club awards a small prize to the members who have the top three times.
The club also awards certificates to all members with a time under 240 seconds.

Write a program that meets the following requirements:

- allows members' times to be input twice and verifies that the inputs match
- sorts the arrays `MemberTime[]` and `MemberName[]` in ascending order of time
- outputs the member names and times of the members with the top three times and identifies them as First, Second and Third
- stores the names of all the members who will receive a certificate in the array `MemberCertificate[]`
- outputs a message stating the number of certificates to be printed.

You must use pseudocode or program code **and** add comments to explain how your code works.

You do **not** need to initialise the data in the array `MemberName[]`

You do **not** need to declare any arrays or variables; you may assume that this has already been done.

All inputs and outputs must contain suitable messages.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [15]

11 A one-dimensional (1D) array `Rooms[]` contains the names of up to 20 rooms in a house.
A two-dimensional (2D) array `Dimensions[]` is used to store the length, width and area of each room.

The position of any room's data is the same in both arrays. For example, the data in index 5 of `Dimensions[]` belongs to the room in index 5 of `Rooms[]`

The variable `Number` stores the number of rooms for which data is to be input. There must be at least 3 rooms but no more than 20.

Write a program that meets the following requirements:

- allows the number of rooms for which data is required to be input, stored and validated
- allows the name of the room and the length and width of the room, in metres, to be entered and stored
- allows the area of each room to be calculated as length multiplied by width and stored as square metres rounded to two decimal places
- calculates the average size of all the rooms by area, in square metres, rounded to two decimal places
- finds the largest room and smallest room by area
- outputs the names of all rooms with their dimensions and area
- outputs the names of the largest room and smallest room by area
- outputs the total area of the house and the average size of all the rooms by area.

You must use pseudocode or program code **and** add comments to explain how your code works.

You do **not** need to declare any arrays or variables; you may assume that this has already been done.

All inputs and outputs must contain suitable messages.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [15]

12 Members of a litter picking group complete a litter pick every month. Members' names are stored in one-dimensional (1D) array `PickerName[]`

Each member stores the weight of the litter they have picked in another one-dimensional (1D) array `PickedWeight[]`
The weights are in kilograms with one decimal place, for example 8.4

The position of each member's data in the two arrays is the same. For example, the member stored at index 10 in `PickerName[]` and at index 10 in `PickedWeight[]` is the same.

Every month, there is a small prize awarded to the members of the group who have the two heaviest weights. Certificates are awarded to all members with a pick weight of over three kilograms.

Write a program that meets the following requirements:

- allows the weight of members' picks to be input and validated
- sorts the arrays `PickedWeight[]` and `PickerName[]` in descending order of weight
- outputs the member names and the pick weights of the members with the two heaviest picks and identifies them as "Best in Group" and "Second best in Group"
- stores the names of all the members who will receive a certificate in the array `PickerCertificate[]`
- outputs a message stating the number of certificates to be printed.

You must use pseudocode or program code **and** add comments to explain how your code works.

You do **not** need to declare any arrays or variables; you may assume that this has already been done.

All inputs and outputs must contain suitable messages.

You do **not** need to initialise the data in the array `PickerName[]`

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [15]