Answers to Algorithms, Programming and Logic Workbook

7 Algorithm design and problem solving

1 Three from:

- analysis – a process of investigation, leading to the specification of what a program is required to do
- design – uses the program specification from the analysis stage to show how the program should be developed
- coding – the writing of the program or suite of programs
- testing – the completed program is run with test data to make sure that all of the modules interact correctly, and the program as a whole works as intended in all circumstances

2 **Abstraction**

Stage Analysis

Use to identify the key elements required, discarding any unnecessary details and information that is not required

Decomposition

Stage Analysis

Use to break down a complex problem into smaller parts, repeating this until each part can be solved easily

Structure diagram

Stage Design

Use a hierarchical diagram that shows how a computer system can be repeatedly divided into sub-systems

3 a data

hardware

communications

b Three from:

- input
- process
- output
- storage

4 Structure diagrams are used to show top-down design in a diagrammatic form.

Flowcharts show diagrammatically the steps required to complete a task and the order that they are to be performed.

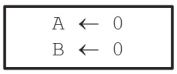
Pseudocode describes what an algorithm does by using English key words that are very similar to those used in a high-level programming language.

5 Structure diagrams are used to show the **top-down** design of a computer **program** in a **diagrammatic** form.

A structure diagram shows the design of a computer program in a **hierarchical** way with each **level** giving a more detailed **breakdown** of the **system** into **sub-systems**.

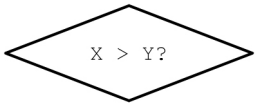
6 a

Terminator flowchart symbols are used at the beginning and end of each flowchart.



Process flowchart symbols are used to show actions.

Input and output symbols – the same flowchart symbol is used to show input of data and output of information.



Decision flowchart symbols are used to decide which action is to be taken next; these can be used for selection and repetition/iteration.

b Flowchart flow lines use arrows to show the direction of flow through the chart, which is usually, but not always, top to bottom and left to right.

7 Assignment: The single variable on the left of the ← is assigned the value of the expression on the right. For example:

```
Total ← Total + SubTotal
```

Conditional: When different actions are performed by an algorithm according to the values of the variables, conditional statements can be used to decide which action should be taken.

```
IF Age > 70
  THEN
    OUTPUT "Senior"
  ELSE
    OUTPUT "Adult or Junior"
ENDIF
```

Iterative: When some actions performed as part of an algorithm need repeating this is called iteration. Loop structures are used to perform the iteration.

```
FOR Counter ← 1 TO 10
    OUTPUT "*"
NEXT Counter
```

8 a Loop 1: `FOR .. TO .. NEXT:`

```
FOR Counter ← 0 TO 19
    OUTPUT Student[Counter]
NEXT Counter
```

Loop 2: `REPEAT .. UNTIL`

```
Counter ← 0
REPEAT
    OUTPUT Student[Counter]
    Counter ← Counter + 1
UNTIL Counter > 19
```

Loop 3: `WHILE .. DO .. ENDWHILE`

```
Counter ← 0
WHILE Counter < 20 DO
    Student[Counter]
    Counter ← Counter + 1
ENDWHILE
```

b i Conditional Statements 1: IF statements

```
OUTPUT "Please enter your name "
INPUT Name
IF Name = "Alice"
  THEN
    OUTPUT "Welcome Alice "
ENDIF
IF Name = "Abid"
  THEN
    OUTPUT "Welcome Abid "
ENDIF
IF Name = "Dinesh"
  THEN
    OUTPUT "Welcome Dinesh "
ENDIF
IF Name = "Daisy"
  THEN
    OUTPUT "Welcome Daisy "
ENDIF
IF Name = "Zak"
  THEN
    OUTPUT "Welcome Zak "
ENDIF
```

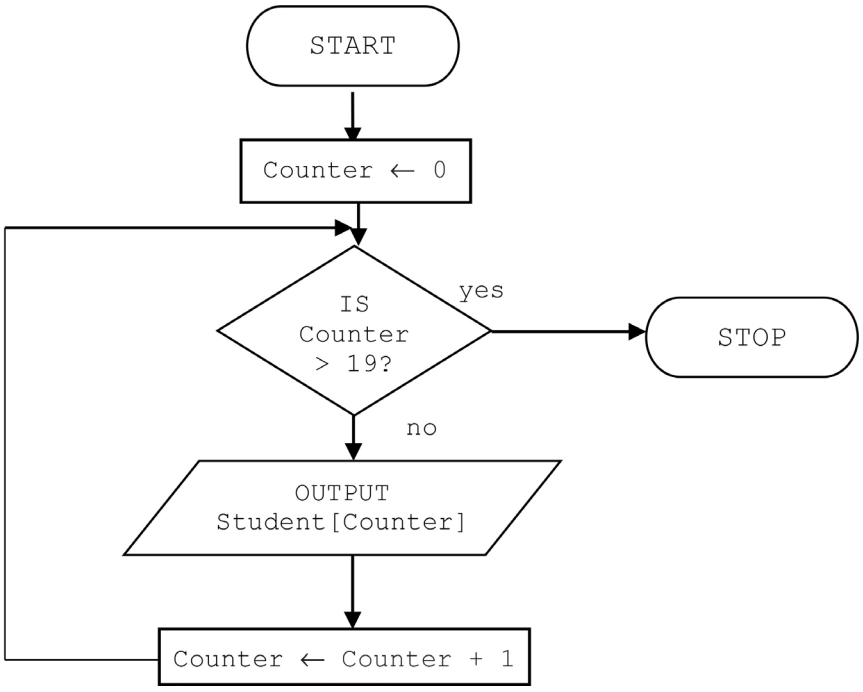
```
IF Name <> "Alice" AND Name <> "Abid" AND Name <> "Dinesh"
  AND Name <> "Daisy" <> AND Name <> "Zak"
  THEN
    OUTPUT "You are not welcome "
ENDIF
```

Conditional Statements 2: CASE statements

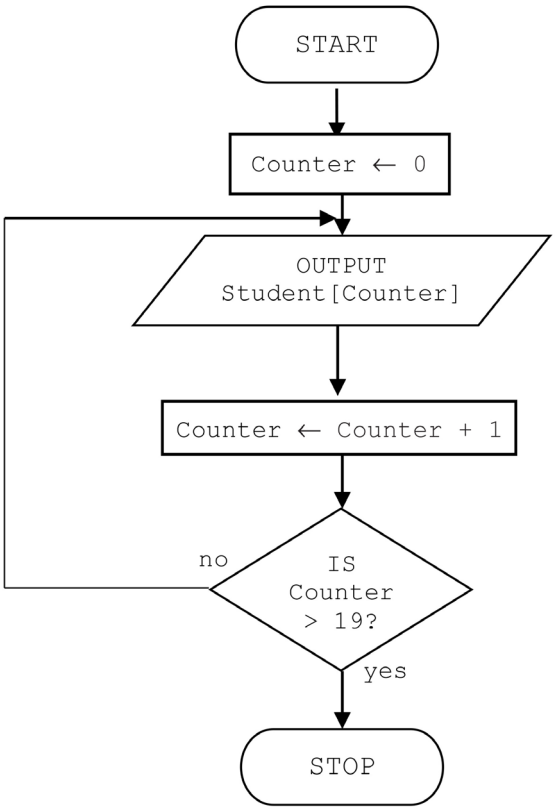
```
OUTPUT "Please enter your name "
INPUT Name
CASE OF Name
  "Alice"   : OUTPUT "Welcome Alice "
  "Abid"    : OUTPUT "Welcome Abid "
  "Dinesh"  : OUTPUT "Welcome Dinesh "
  "Daisy"   : OUTPUT "Welcome Daisy "
  "Zak"     : OUTPUT "Welcome Zak "
  OTHERWISE OUTPUT "You are not welcome "
ENDCASE
```

ii
- The most suitable conditional statement to use is `CASE` as the code is considerably shorter to write …
- … and clearer to read and understand.

c i Pre-condition loop



Post-condition loop



ii Pre-condition loop WHILE .. DO .. ENDWHILE

Post-condition loop REPEAT .. UNTIL

d Loop 1

```
FOR .. TO .. NEXT
```

Python

```
Student = ["Alice", "Jan", "Lim", "Ian", "Min", "Sue", "Tim",
"Dee", "Avril", "Ron", "Fred", "Idris", "May", "Susie", "Tony",
"Dom", "Ted", "Davy", "Chen", "Bao"]

Counter = 0
for Counter in range(20):
    print(Student[Counter])
print()
```

Visual Basic

```
Dim Student = New String() {"Alice", "Jan", "Lim", "Ian", "Min",
"Sue", "Tim", "Dee", "Avril", "Ron", "Fred", "Idris", "May",
"Susie", "Tony", "Dom", "Ted", "Davy", "Chen", "Bao"}

Dim Counter As Integer

'FOR .. TO .. NEXT
For Counter = 0 To 19
    Console.WriteLine(Student(Counter))
Next
Console.WriteLine()
```

Java

```
String[] Student ={"Alice", "Jan", "Lim", "Ian", "Min", "Sue",
"Tim", "Dee", "Avril", "Ron", "Fred", "Idris", "May", "Susie",
"Tony", "Dom", "Ted", "Davy", "Chen", "Bao"};
int Counter;

// FOR .. NEXT
    for (Counter = 0; Counter <= 19; Counter ++) {
        System.out.println(Student[Counter]);
    }
    System.out.println();
```

Loop 2

```
WHILE .. DO .. ENDWHILE
```

Python

```
Student = ["Alice", "Jan", "Lim", "Ian", "Min", "Sue", "Tim",
"Dee", "Avril", "Ron", "Fred", "Idris", "May", "Susie", "Tony",
"Dom", "Ted", "Davy", "Chen", "Bao"]

Counter = 0
while Counter < 20:
    print(Student[Counter])
    Counter = Counter + 1
print()
```

Visual Basic

```
Dim Student = New String() {"Alice", "Jan", "Lim", "Ian", "Min",
"Sue", "Tim", "Dee", "Avril", "Ron", "Fred", "Idris", "May",
"Susie", "Tony", "Dom", "Ted", "Davy", "Chen", "Bao"}

Dim Counter As Integer
'WHILE .. DO .. ENDWHILE
Counter = 0
While Counter < 20
    Console.WriteLine(Student(Counter))
    Counter = Counter + 1
End While
Console.WriteLine()
```

Java

```
String[] Student ={"Alice", "Jan", "Lim", "Ian", "Min", "Sue",
"Tim", "Dee", "Avril", "Ron", "Fred", "Idris", "May", "Susie",
"Tony", "Dom", "Ted", "Davy", "Chen", "Bao"};

int Counter;
// WHILE .. DO .. ENDWHILE
Counter = 0;
while (Counter <= 19){
    System.out.println(Student[Counter]);
    Counter ++;
}
System.out.println();
```

Loop 3

```
REPEAT .. UNTIL
```

Python Not available

Visual Basic

```
Dim Student = New String() {"Alice", "Jan", "Lim", "Ian", "Min",
"Sue", "Tim", "Dee", "Avril", "Ron", "Fred", "Idris", "May",
"Susie", "Tony", "Dom", "Ted", "Davy", "Chen", "Bao"}

Dim Counter As Integer
'REPEAT .. UNTIL
Counter = 0
Do
    Console.WriteLine(Student(Counter))
    Counter = Counter + 1
Loop Until Counter = 19
Console.WriteLine()
```

Java

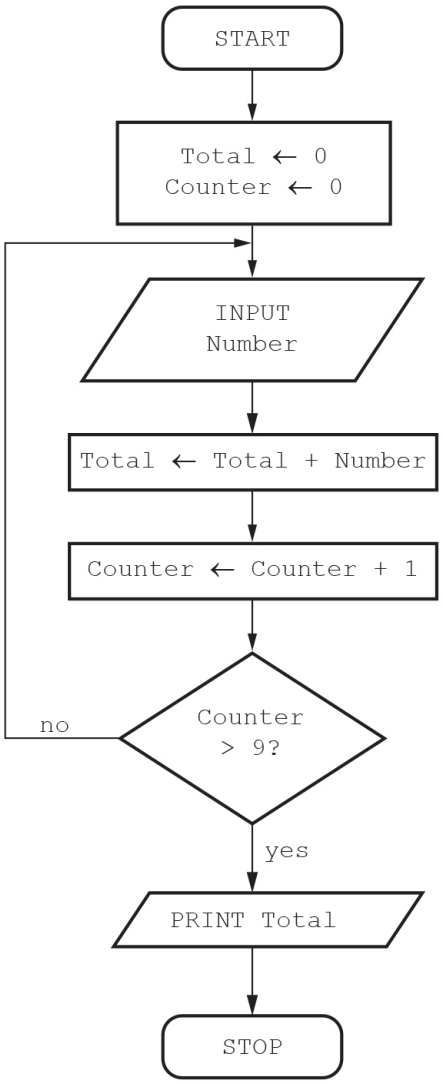
```
String[] Student ={"Alice", "Jan", "Lim", "Ian", "Min", "Sue",
"Tim", "Dee", "Avril", "Ron", "Fred", "Idris", "May", "Susie",
"Tony", "Dom", "Ted", "Davy", "Chen", "Bao"};

int Counter;
// REPEAT .. UNTIL
Counter = 0;
do {
    System.out.println(Student[Counter]);
    Counter ++;
}
while (Counter <= 19);
System.out.println();
```

9 a

```
DECLARE Number, Total : REAL
DECLARE Counter : INTEGER
Total ← 0
FOR Counter ← 1 TO 10
  OUTPUT "Please enter next Number"
  INPUT Number
  Total ← Total + Number
NEXT Counter
OUTPUT "The total of the 10 numbers input is ", Total
```

b



10 a input – the number of guesses allowed

input – a guess for a word

check – checks if a guess is correct

check – checks how many guesses are left

output – Message(s) to show if a guess is correct or not

b
- To play a word guessing game …
- … allowing the player to decide how many guesses to take.

c
```
// The code below represents the guessing algorithm.
// Assume that the word to guess, W, has already been
// declared and defined.
DECLARE N : INTEGER
DECLARE G : STRING
DECLARE Finish : BOOLEAN
Finish ← FALSE
OUTPUT "How many guesses would you like?"
INPUT N
REPEAT
    OUTPUT "Please enter your guess: "
    INPUT G
    IF G = W
      THEN
        OUTPUT "Well done you guessed the word!"
        Finish ← TRUE
      ELSE
        OUTPUT "You failed to guess the word!"
        N ← N - 1
    ENDIF
UNTIL Finish OR N = 0
```

11 Standard methods of solution are used in the design of **algorithms**. These include adding in a new value every time an action occurs, for example, awarding a mark to each student – this is called **totalling**. When divided by the number of times this occurs, this gives the **average**. Keeping a record of the number of times an action occurs is called **counting**. Finding the largest, **maximum**, and smallest, **minimum**, are also standard methods.

In order to look for an item in a list a **search** is used. The method you need to know for IGCSE Computer Science is to inspect each item in the **list** in turn to see if it is the one required. This is called a **linear search**.

To put a list in order a **sort** is used. The method you need to know for IGCSE Computer Science is called a **bubble sort**.

12 a
- Each element in a list is compared with the next element and swapped if the elements are in the wrong order …
- … starting from the first element and finishing with the next-to-last element.

b
```
First ← 1
Last ← 10
REPEAT
    Swap ← FALSE
    FOR Index ← First TO Last - 1
      IF Names[Index] > Names[Index + 1]
        THEN
          Temp ← Names[index]
          Names[index] ← Names[index + 1]
          Names[index + 1] ← Temp
          Swap ← TRUE
      ENDIF
    NEXT Index
    Last ← Last - 1
UNTIL (NOT Swap) OR (Last = 1)
```

13 a i Validation is appropriate to check that only reasonable data is accepted when data is entered.

ii Verification is appropriate to check that the data does not change as it is being entered.

b i range check

presence check

ii double entry

visual check

14 a
- Test data is a set of values used to determine whether a solution is working as it should …
- … the values are chosen to test a full range of operating conditions.

b i normal data – a set of test data used that the program would normally be expected to work with

abnormal/erroneous data – test data chosen to be rejected by the solution as not suitable, if the solution is working properly

extreme data – the largest and smallest values that normal data can take

ii 99 inside the boundary

100 outside the boundary

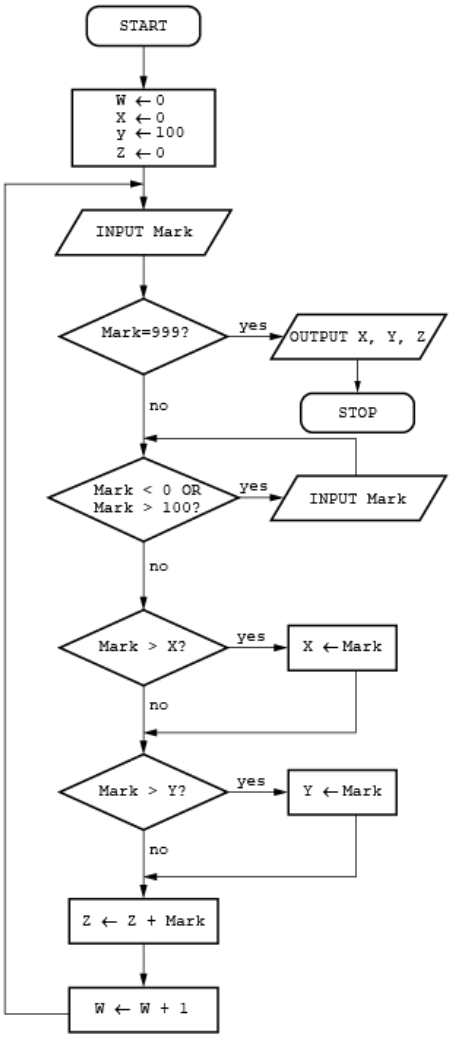
15 a

W	X	Y	Z	Mark	OUTPUT
0	0	100	0		
1	78	78	78	78	
2		34	112	34	
3		22	134	22	
				-4	
4	98		232	98	
5		16	248	16	
				734	
6			336	88	
				999	
					98 16 336

b

- selects the maximum and minimum values input between 0 and 100 …
- … calculates the total of these valid input values …
- … and outputs the maximum, minimum and the total.

c



d

```
NumberOfMarks ← 0
HighMark ← 0
LowMark ← 100
Total ← 0
REPEAT
    OUTPUT "Enter 999 to finish"
    OUTPUT "Marks must be in the range 0-100, enter next
      mark"
INPUT Mark
IF Mark <> 999
  THEN
    REPEAT
        IF Mark <0 OR Mark >100
          THEN
            OUTPUT "Your mark is out of range,
              please re-enter the mark"
            INPUT Mark
        ENDIF
    UNTIL Mark >=0 AND Mark <=100
    IF Mark > HighMark
      THEN
        HighMark ← Mark
    ENDIF
    IF Mark < LowMark
      THEN
        LowMark ← Mark
    ENDIF
  Total ← Total + Mark
  NumberOfMarks ← NumberOfMarks + 1
ENDIF
UNTIL Mark = 999
OUTPUT "The highest mark is ", HighMark
OUTPUT "The lowest mark is ", LowMark
OUTPUT "The total of all the marks is ", Total
```

e Example test data could be: Test data: −1, 0, 100, 101, −5, 21, 999

This has been chosen so that there is mixture of normal, abnormal, extreme and boundary data. In this case:

Normal data: **21**, **999** : 999 has been chosen to ensure that the method used to stop the program works and should be included in test data; but any other value in the range 1–99 could also be used.

Abnormal data: **−5** : but any other data larger than 101, less than −1, or of the wrong data type (e.g. twenty one) could be used

Extreme data: **0**, **100** : these are the only choice of extreme data for this example

Boundary data: **−1**, **0**, **100**, **101** : these are the only choice of boundary data for the main data range (although some students might also want to test the boundary for the 'quit' option, i.e. 998, 999, 1000).

Trace table:

NumberOf Marks	HighMark	Low Mark	Total	Mark	OUTPUT
0	0	100	0		Enter 999 to finish Marks must be in the range 0-100, enter next mark
				−1	Your mark is out of range, please re-enter the mark
1		0	0	0	Enter 999 to finish Marks must be in the range 0-100, enter next mark
2	100		100	100	Enter 999 to finish Marks must be in the range 0-100, enter next mark
				101	Your mark is out of range, please re-enter the mark
				−5	Your mark is out of range, please re-enter the mark
3			121	21	Enter 999 to finish Marks must be in the range 0-100, enter next mark
				999	The Highest mark is 100 The lowest mark is 0 The total of all the marks is 121

16 a `OUTPUT T` should be `INPUT T`

`T < 30` should be `T < 20`

No flowline out of the `OUTPUT "Too Low"` box.

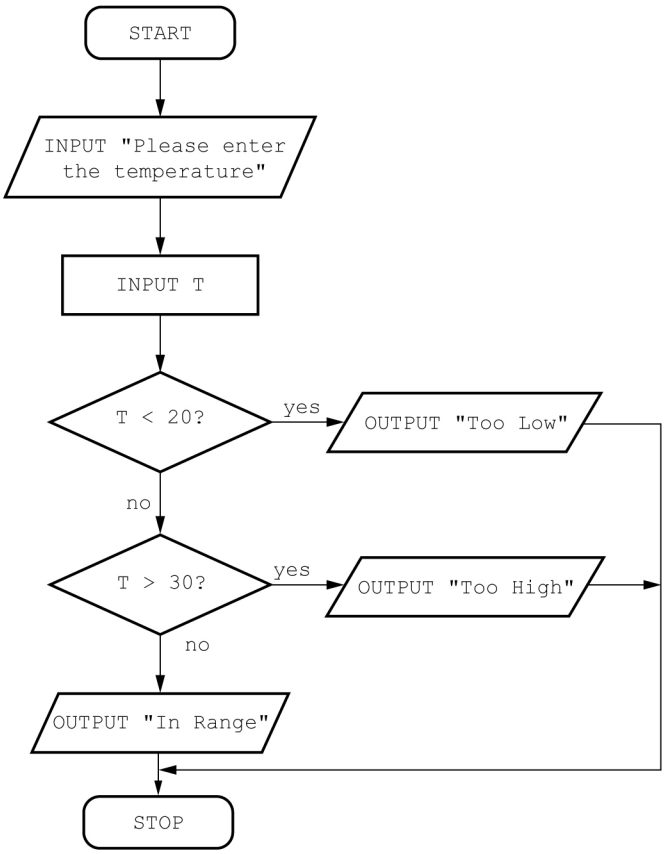
`T > 20` should be `T > 30`

Flowlines from the decision box are not labelled.

The arrow on the horizontal flowline is at the wrong end and points in the wrong direction.

No flowline out of the `OUTPUT "Too High"` box.

b



8 Programming

1 Three from:

- integer: a positive or negative whole number that can be used with mathematical operators
- real: a positive or negative number with a fractional part that can be used with mathematical operators
- Boolean: a variable or constant that can have only two values `TRUE` or `FALSE`
- char: a single alphanumeric character
- string: multiple alphanumeric characters

2 a Python

```
Number1 = int(input("Please enter the first whole number "))
Number2 = int(input("Please enter the second whole number "))
Number3 = int(input("Please enter the third whole number "))
```

Visual Basic

```
Console.Write("Please enter first whole number ")
Number1 = Integer.Parse(Console.ReadLine())
Console.Write("Please enter second whole number ")
Number2 = Integer.Parse(Console.ReadLine())
Console.Write("Please enter third whole number ")
Number3 = Integer.Parse(Console.ReadLine())
```

Java

```
System.out.print("Please enter the first whole number ");
Number1 = myObj.nextInt();
System.out.print("Please enter the second whole number ");
Number2 = myObj.nextInt();
System.out.print("Please enter the third whole number ");
Number3 = myObj.nextInt();
```

b Python

```
Number1 = 10
Number2 = 20
Number3 = 30
```

Visual Basic

```
Number1 = 10
Number2 = 20
Number3 = 30
```

Java

```
Number1 = 10;
Number2 = 20;
Number3 = 30;
```

c Python

```
print("Integer values assigned and stored", Number1, Number2,
Number3)
```

Visual Basic

```
Console.WriteLine("Integer values assigned and stored " &
Number1 & " " & Number2 & " " & Number3)
```

Java

```
System.out.println("Integer values assigned and stored " +
Number1 + " " + Number2 + " " + Number3);
```

d **Python**

```
Number1 = int(input("Please enter the first whole number "))
Number2 = int(input("Please enter the second whole number "))
Number3 = int(input("Please enter the third whole number "))

print("Integer values entered", Number1, Number2, Number3)

Number1 = 10
Number2 = 20
Number3 = 30

print("Integer values assigned and stored" , Number1,
Number2, Number3)
```

Visual Basic

```
Module Module1

    Sub Main()
        Dim Number1, Number2, Number3 As Integer
        Console.Write("Please enter first whole number ")
        Number1 = Integer.Parse(Console.ReadLine())
        Console.Write("Please enter second whole number ")
        Number2 = Integer.Parse(Console.ReadLine())
        Console.Write("Please enter third whole number ")
        Number3 = Integer.Parse(Console.ReadLine())

        Console.WriteLine("Integer values entered " & Number1
        & " " & Number2 & " " & Number3)

        Number1 = 10
        Number2 = 20
        Number3 = 30

        Console.WriteLine("Integer values assigned and stored
        " & Number1 & " " & Number2 & " " & Number3)

        Console.ReadKey()
    End Sub

End Module
```

Java

```
import java.util.Scanner;
class Q2 Java
{
   public static void main(String args[])
   {
       Scanner myObj = new Scanner(System.in);

       int Number1, Number2, Number3;

       System.out.print("Please enter the first whole number ");
       Number1 = myObj.nextInt();
       System.out.print("Please enter the second whole number ");
       Number2 = myObj.nextInt();
       System.out.print("Please enter the third whole number ");
       Number3 = myObj.nextInt();

       System.out.println("Integer values entered  " +
       Number1 + " " + Number2 +" " + Number3);

       Number1 = 10;
       Number2 = 20;
       Number3 = 30;

       System.out.println("Integer values assigned and
       stored" + Number1 + " " + Number2 + " " + Number3);
    }
}
```

3 a **Python**

```
weight = 0
while weight < 0.5 or weight > 5.0:
    weight = float(input("Please enter the weight of your
    parcel "))
```

Visual Basic

```
weight = 0
While (weight < 0.5) Or (weight > 5.0)
    Console.Write("Please enter the weight of your parcel  ")
    weight = Decimal.Parse(Console.ReadLine())
End While
```

Java

```
do {
  System.out.print("Please enter the weight of your parcel ");
  weight = myObj.nextDouble();
}
while (weight < 0.5 || weight > 5.0);
```

b **Python**

```
print("          Options")
print(" 1 Guaranteed next day delivery before noon")
print(" 2 Guaranteed next day delivery")
print(" 3 24-hour delivery")
print(" 4 48-hour delivery")
print(" 5 3-5 days delivery")
option = int(input("Please enter your chosen option "))

if option == 1:
    # option 1 code
elif option == 2:
    # option 2 code
elif option == 3:
    # option 3 code
elif option == 4:
    # option 4 code
elif option == 5:
    # option 5 code
else:
     print("Incorrect option chosen")
```

Visual Basic

```
Console.WriteLine("          Options")
Console.WriteLine(" 1 Guaranteed next day delivery before
noon")
Console.WriteLine(" 2 Guaranteed next day delivery")
Console.WriteLine(" 3 24-hour delivery")
Console.WriteLine(" 4 48-hour delivery")
Console.WriteLine(" 5 3-5 days delivery")

Console.Write ("Please enter your chosen option ")
opt = Integer.Parse(Console.ReadLine())
```

```
Select Case opt
     Case 1
        'option 1 code
     Case 2
        'option 2 code
     Case 3
         'option 3 code
     Case 4
         'option 4 code
     Case 5
         'option 5 code
     Case Else
         Console.WriteLine("Incorrect option chosen")
End Select
```

Java

```
System.out.println("          Options");
System.out.println(" 1 Guaranteed next day delivery before
noon");
System.out.println(" 2 Guaranteed next day delivery");
System.out.println(" 3 24-hour delivery");
System.out.println(" 4 48-hour delivery");
System.out.println(" 5 3-5 days delivery");

System.out.print("Please enter your chosen option ");
option = myObj.nextInt();
switch (option) {
  case 1:
    //option 1 code ;
    break;
  case 2:
    //option 2 code;
    break;
  case 3:
    //option 3 code;
    break;
  case 4:
    //option 4 code;
    break;
```

```
    case 5:
      //option 5 code;
      break;
    default:
      System.out.println("Incorrect option chosen");
  }
```

c **Python**

```
weight = 0
while weight < 0.5 or weight > 5.0:
    weight = float(input("Please enter the weight of your
    parcel "))

print("          Options")
print(" 1 Guaranteed next day delivery before noon")
print(" 2 Guaranteed next day delivery")
print(" 3 24-hour delivery")
print(" 4 48-hour delivery")
print(" 5 3-5 days delivery")
option = int(input("Please enter your chosen option "))

if option == 1:
    cost = weight * 10 + 1
elif option == 2:
    cost = weight * 10
elif option == 3:
    cost = 5
elif option == 4:
    cost = 4
elif option == 5:
    cost = 3
else:
    print ("Incorrect option chosen")
    cost = 0

print ("Your cost is ", cost)
```

Visual Basic

```
Module Module1

    Sub Main()
        Dim weight, cost As Decimal
        Dim opt As Integer
        weight = 0
        While (weight < 0.5) Or (weight > 5.0)
           Console.Write("Please enter the weight of your
           parcel ")
           weight = Decimal.Parse(Console.ReadLine())
        End While

        Console.WriteLine("          Options")
        Console.WriteLine(" 1 Guaranteed next day delivery
        before noon")
        Console.WriteLine(" 2 Guaranteed next day delivery")
        Console.WriteLine(" 3 24-hour delivery")
        Console.WriteLine(" 4 48-hour delivery")
        Console.WriteLine(" 5 3-5 days delivery")

        Console.Write("Please enter your chosen option ")
        opt = Integer.Parse(Console.ReadLine())

        Select Case opt
            Case 1
                cost = weight * 10 + 1
            Case 2
                cost = weight * 10
            Case 3
                cost = 5
            Case 4
                cost = 4
            Case 5
                cost = 3
            Case Else
                Console.WriteLine("Incorrect option chosen")
                cost = 0
        End Select
```

```
        Console.WriteLine("Your cost is " & cost)

        Console.ReadKey()

    End Sub

End Module
```

Java

```
import java.util.Scanner;
class Q3Java
{
   public static void main(String args[])
   {
       Scanner myObj = new Scanner(System.in);
       double weight, cost;
       int option;

       do {
       System.out.print("Please enter the weight of your
       parcel ");
       weight = myObj.nextDouble();
       }
       while (weight < 0.5 || weight > 5.0);

       System.out.println("          Options");
       System.out.println(" 1 Guaranteed next day delivery
       before noon");
       System.out.println(" 2 Guaranteed next day delivery");
       System.out.println(" 3 24-hour delivery");
       System.out.println(" 4 48-hour delivery");
       System.out.println(" 5 3-5 days delivery");

       System.out.print("Please enter your chosen option ");
       option = myObj.nextInt();
       switch (option) {
         case 1:
           cost = weight * 10 + 1;
           break;
```

```
            case 2:
              cost = weight * 10;
              break;
            case 3:
              cost = 5;
              break;
            case 4:
              cost = 4;
              break;
            case 5:
              cost = 3;
              break;
            default:
              System.out.println("Incorrect option chosen");
              cost = 0;
              }
          System.out.println("Your cost is " + cost);
        }
    }
```

d

Test data	Expected output	Actual output
0.4	Error message	
5.1	Error message	
−1	Error message	
0.5 Option 1	Your cost is 6	
0.5 Option 2	Your cost is 5	
0.5 Option 3	Your cost is 5	
5 Option 1	Your cost is 51	
5 Option 2	Your cost is 50	
5 Option 4	Your cost is 4	
1 Option 5	Your cost is 3	
1 Option 7	Incorrect option chosen	

4 a Counting: keeping a count of the number of times an action is performed:

```
Counter = Counter + 1
```

b Iteration: a section of programming code can be repeated under certain conditions.

For example, a pre-condition loop:

Python

```
while TotalWeight < 100:
        TotalWeight = TotalWeight + Weight
```

Visual Basic

```
While TotalWeight < 100
      TotalWeight = TotalWeight + Weight
End While
```

Java

```
while (TotalWeight < 100)
{
  TotalWeight = TotalWeight + Weight;
}
```

c Selection: allowing the selection of different paths through the steps of a program:

Python

```
if Age > 17:
    print("You are an adult")
```

Visual Basic

```
If Age > 17 Then
   Console.WriteLine("You are an adult")
End If
```

Java

```
if (Age > 17) {
  System.out.println ("You are an adult");
}
```

d Sequence – the order that steps in a program are executed.

For example: `TotalWeight` and `Weight` must be assigned values before the WHILE loop is executed.

Python

```
TotalWeight = 0
while TotalWeight < 100:
     Weight = float(input("Enter the next weight: "))
     TotalWeight = TotalWeight + Weight
```

Visual Basic

```
Dim Weight, TotalWeight As Decimal
TotalWeight = 0

While TotalWeight < 100
    Console.WriteLine("Please enter the weight of your parcel ")
    Weight = Decimal.Parse(Console.ReadLine())
    TotalWeight = TotalWeight + Weight
End While
```

Java

```
import java.util.Scanner;
class Q4{
      public static void main(String args[])
      {
            double TotalWeight = 0;
            double Weight;
            Scanner myObj = new Scanner(System.in);

            while (TotalWeight < 100)
            {
                  System.out.print("Please enter the weight of
                  your parcel ");
                  Weight = myObj.nextDouble();
                  TotalWeight = TotalWeight + Weight;
            }
      }
}
```

e String handling – strings are used to store text. Every string contains a number of characters, from an empty string which has no characters stored to a maximum number specified by the programming language. The characters in a string can be labelled by position number. The first character in a string can be in position zero or position one, depending on the language.

For example, finding the number of characters in a string:

Python

```
len(MyString)
```

Visual Basic

```
MyString.Length()
```

Java

```
MyString.length();
```

5

```
// Algorithm to input ten positive numbers and total them,
// output the total and then average
   DECLARE A, B, C, D : INTEGER
   DECLARE D : REAL
   A ← 0
   B ← 0

   REPEAT
       REPEAT
           OUTPUT "Please enter a positive number "
           INPUT C
       UNTIL C > 0
       A ← A + C
       B ← B + 1
   UNTIL B = 10
   D ← A / B
   OUTPUT A, D
```

6 a **Python**

```
# Python program to total 10 numbers and find the average
Total = 0
Counter = 0  # Total and Counter initialised to zero

while Counter < 10:
    Number = int(input("Please enter a positive number "))
    while Number <= 0:  # error trapping for negative numbers
        Number = int(input("Your number was zero or negative, please try
        again"))
    Total = Total + Number
    Counter = Counter + 1  # updating Total and Counter

Average = Total / Counter  # calculating the Average

print(" The total is ", Total, " the average is ", Average)
# outputting the results
```

Visual Basic

```
'Visual Basic program to total 10 numbers and find the average

Module Module1

    Sub Main()
        Dim Total, Number, Counter As Integer
        Dim Average As Decimal ' variables declared

        Total = 0
        Counter = 0 ' Total and Counter initialised To zero

        Do
            Console.Write("Please enter a positive number ")
            Number = Int(Console.ReadLine())

            While (Number <= 0) 'error trapping for negative numbers
              Console.Write("Your number was zero or negative. Please try
              again ")
              Number = Int(Console.ReadLine())
            End While

            Total = Total + Number
            Counter = Counter + 1  'updating Total And Counter
        Loop Until Counter = 10

        Average = Total / Counter  ' calculating the Average

        Console.WriteLine(" The total is " & Total & " the average is " &
        Average) ' outputting the results

        Console.ReadLine()

    End Sub

End Module
```

Java

```
// Java program to total 10 numbers and find the average
import java.util.Scanner;
class Q6Java
{
   public static void main(String args[])
   {
       Scanner myObj = new Scanner(System.in);

       int Counter, Number;
       double Average Total; //variables declared , Total must be double
       otherwise the division will be truncated

       Total = 0;
       Counter = 0; //Total and Counter initialised To zero

       do {

            System.out.print("Please enter a positive number ");
            Number = myObj.nextInt();

            while (Number <= 0){

                System.out.print("Your number was zero or negative. Please
                try again ");
                Number = myObj.nextInt(); //error trapping for negative
                numbers
            }

            Total = Total + Number;
            Counter = Counter + 1; //updating Total And Counter
        }
        while (Counter < 10);

        Average = Total / Counter ; // calculating the Average

        System.out.println(" The total is " + Total + " the average is " +
        Average);//outputting the results
    }
}
```

b

Total	Number	Counter	Average	OUTPUT
0		0		
4	4	1		Please enter a positive number
7	3	2		Please enter a positive number
14	7	3		Please enter a positive number
20	6	4		Please enter a positive number
	0			Your number was negative or zero, please try again
	-3			Your number was negative or zero, please try again
29	9	5		Please enter a positive number
30	1	6		Please enter a positive number
38	8	7		Please enter a positive number
43	5	8		Please enter a positive number
45	2	9		Please enter a positive number
50	5	10		Please enter a positive number
			5.0	The total is 50 the average is 5.0

7 a

Python

```
Student = ["Alice", "Jan", "Lim", "Ian", "Min", "Sue", "Tim",
"Dee"]
for Counter in range(8):
    print(Student[Counter])
print()

Counter = 0
while Counter < 8:
    print(Student[Counter])
    Counter = Counter + 1
print()
```

Visual Basic

```
For Counter = 0 To 7
    Console.WriteLine(Student(Counter))
Next
Console.WriteLine()
```

```
Counter = 0
Do
    Console.WriteLine(Student(Counter))
    Counter = Counter + 1
Loop Until Counter = 8

Console.WriteLine()
```

Java

```
for (Counter = 0; Counter <= 7; Counter++) {
     System.out.println(Student[Counter]);
}
System.out.println()

Counter = 0;
while (Counter <= 7){
      System.out.println(Student[Counter]);
      Counter++;
}

System.out.println()
```

b **Python**

```
NumberOfStudents = len(Student)

Counter = 0
while Counter < NumberOfStudents:
    print(Student[Counter])
    Counter = Counter + 1
print("Number of students", NumberOfStudents)
```

Visual Basic

```
NumberOfStudents = Student.Length
Counter = 0
Do
    Console.WriteLine(Student(Counter))
    Counter = Counter + 1
Loop Until Counter = NumberOfStudents

Console.WriteLine("Number of students " & NumberOfStudents)
```

Java

```
NumberOfStudents = Student.length;
Counter = 0;
while (Counter < NumberOfStudents){
    System.out.println("Number of students " + NumberOfStudents);
    Counter++;
}
```

8 **a** Find the length: `LENGTH(MyString)`

Convert to upper case: `UCASE(MyString)`

Convert to lower case: `LCASE(MyString)`

Find the first character: `SUBSTRING(MyString, 1, 1)`

b **Python**

```
MyString = "Test String"
Length = len(MyString)
FirstChar = MyString[0:1]
UpperCase = MyString.upper()
LowerCase = MyString.lower()

print("Length of string ", Length)
print("First character of string ", FirstChar)
print("String in upper case ", UpperCase)
print("String in lower case ", LowerCase)
```

Visual Basic

```
Module Module1

    Sub Main()

        Dim MyString, FirstChar, UpperCase, LowerCase As String
        Dim Length As Integer
        MyString = "Test String"
        Length = MyString.Length()
        FirstChar = MyString.Substring(0, 1)
        UpperCase = UCase(MyString)
        LowerCase = LCase(MyString)

        Console.WriteLine("Length of string " & Length)
        Console.WriteLine("First character of string " & FirstChar)
        Console.WriteLine("String in upper case " & UpperCase)
        Console.WriteLine("String in lower case " & LowerCase)

        Console.ReadKey()
    End Sub

End Module
```

Java

```
import java.util.*;
class Q8Java
{
   public static void main(String args[])
    {
       Scanner myObj = new Scanner(System.in);


       String MyString, FirstChar, UpperCase, LowerCase;
       int Length;
       MyString = "Test String";
       Length = MyString.length();
       FirstChar = MyString.substring(0, 1);
       UpperCase = MyString.toUpperCase();
       LowerCase = MyString.toLowerCase();

       System.out.println("Length of string " + Length);
       System.out.println("First character of string " +
       FirstChar);
       System.out.println("String in upper case " + UpperCase);
       System.out.println("String in lower case " + LowerCase);

    }
}
```

9 a **Python**

```
A = ((B + C - D * E) ** F) / G
```

Visual Basic

```
A = ((B + C - D * E) ^ F) / G
```

Java

```
A = Math.pow((B + C - D * E), F) / G;
```

b **Python**

```
if (A == B) or (C != D):
    A = 0
elif (A > B) and (C == D):
    B = 0
```

Visual Basic

```
If (A = B) Or (C <> D) Then
    A = 0
Else
    If (A > B) And (C = D) Then
        B = 0
    End If
End If
```

Java

```
if (A == B) or (C != D) {
    A = 0;
}
else {
    if ((A > B) and (C = D)) {
        B = 0;
    }
}
```

c

```
FOR Times = 1 TO 20
    FOR Count = 1 TO 10
        OUTPUT Count
    NEXT Count
NEXT Times
```

d **Python**

```
for Times in range(20):
    for Count in range(10):
            print(Count + 1)
```

Visual Basic

```
Module Module1

    Sub Main()
        Dim Count, Times As Integer
        For Times = 1 To 20
            For Count = 1 To 10
                Console.Write(Count)
            Next
            Console.WriteLine()
        Next

        Console.ReadKey()
    End Sub
End Module
```

Java

```
import java.util.*;
class Q9Java
{
   public static void main(String args[])
    {

       int Count, Times;
       for (Times = 1; Times <= 20; Times++) {
           for (Count = 1; Count <= 10; Count++) {
           System.out.print(Count);
                }
           System.out.println();
       }

    }
}
```

10 Tasks that are repeated many times in an algorithm make use of **procedures** and **functions**. Use of these can reduce the size of a **program**.

Procedures and functions are **defined** once and **called** many times. They can be written with and without **parameters**. They are the variables that store the values passed to a procedure or function.

Functions always **return** a **value** and the value can be used on the right-hand side of an assignment statement.

A variable that can be used in any part of a program is called a **global** variable. A variable that is declared within a procedure or function is called a **local** variable.

11 a Library routines are fully tested and ready to be incorporated into a program …

… they perform many types of task that need to be used regularly.

b 1 MOD

2 DIV

3 ROUND

4 RANDOM

c **Python**

```
import random

Number1 = int(random.random() * 10 + 10)
Number2 = int(random.random() * 10 + 10)
print(Number1, "is the first random integer between 10 and 20")
print(Number2, "is the second random integer between 10 and 20")

AnswerMod = Number1 % Number2
AnswerDiv = Number1 // Number2

print(Number1, "MOD", Number2, "is", AnswerMod)
print(Number1, "DIV", Number2, "is", AnswerDiv)
```

Visual Basic

```
Module Module1

    Sub Main()
        Dim Number1, Number2 As Integer
        Dim AnswerMod, AnswerDiv As Integer
        Randomize()
        Number1 = CInt(Rnd() * 10 + 10)
        Number2 = CInt(Rnd() * 10 + 10)
        Console.WriteLine(Number1 & " is the first random integer
        between 10 and 20")
        Console.WriteLine(Number2 & " is the second random integer
        between 10 and 20")

        AnswerMod = Number1 Mod Number2
        AnswerDiv = Number1 \ Number2

        Console.WriteLine(Number1 & " MOD " & Number2 & " is " &
        AnswerMod)
        Console.WriteLine(Number1 & " DIV " & Number2 & " is " &
        AnswerDiv)


        Console.ReadKey()

    End Sub

End Module
```

Java

```
import java.util.Scanner;
import java.lang.Math;
import java.util.Random;

class Q11Java
{
   public static void main(String args[])
    {
       Scanner myObj = new Scanner(System.in);
       int Number1, Number2;
       Random rand = new Random();
       Number1 = rand.nextInt(10) + 10;
       Number2 = rand.nextInt(10) + 10;

       System.out.println(Number1 + " is the first random integer
       between 10 and 20");
       System.out.println(Number2 + " is the second random
       integer between 10 and 20");
       int AnswerMod = Number1 % Number2;
       int AnswerDiv = Number1 / Number2;

       System.out.println(Number1 + " MOD " + Number2 + " is " +
       AnswerMod);
       System.out.println(Number1 + " DIV " + Number2 + " is " +
       AnswerDiv);

    }
}
```

12 a array: a data structure containing several elements of the same data type; these elements can be accessed using the same identifier name

array dimension: dimension determined by how many index numbers are needed to locate a piece of data, for instance a one-dimensional array only needs one index number, a two dimensional array needs two index numbers, and so on

array index: identifies the position of an element in an array

b

```
DECLARE OXO : ARRAY[3, 3] OF CHAR

DECLARE PROCEDURE ShowGame()
    DECLARE RowIndex, ColumnIndex : INTEGER
    FOR RowIndex ← 1 TO 3
        PRINT OXO[RowIindex,1], OXO[RowIindex,2], OXO[RowIindex,3]
    NEXT
ENDPROCEDURE
```

c **Python**

```
OXO =[["O"," "," "],[" ", "X", " "],[" ", "X", "X"]]
def PrintStateOfGame():
    print(OXO[0])
    print(OXO[1])
    print(OXO[2])

PrintStateOfGame()
```

Visual Basic

```
Module Module1
    Public OXO = New String(2, 2) {{"O", " ", " "}, {" ", "X", " "},
    {" ", "X", "O"}} 'OXO must be declared as a public/global variable

    Sub Main()
        PrintStateOfGame()
        Console.ReadKey()
    End Sub
    Sub PrintStateOfGame()
        Dim RowCounter, ColumnCounter As Integer
        For RowCounter = 0 To 2
            For ColumnCounter = 0 To 2
                Console.Write(OXO(RowCounter, ColumnCounter))
            Next
            Console.WriteLine()
        Next
    End Sub
End Module
```

Java

```
import java.util.*;

class Q12Java

{
    public static String[][] OXO = {{"O", " ", " "}, {" ", "X", " "},
    {" ", "X", "O"}};

    static void PrintStateOfGame()
    {
       int RowCounter, ColumnCounter;
       for (RowCounter = 0; RowCounter <= 2; RowCounter++){
           for (ColumnCounter = 0; ColumnCounter <= 2; ColumnCounter++){
               System.out.print(OXO[RowCounter][ColumnCounter]);
            }
       System.out.println();
    }
```

```
}

    public static void main(String args[])
    {
              PrintStateOfGame();
    }
```

13 a Data that will be required again will be lost if it is only in RAM when the computer is switched off …

… if data is stored in a file, it can be accessed by the same or another program at a later date.

b i

```
DECLARE Text : STRING
DECLARE MyFile : STRING


MyFile ← "MyFile.txt"
Text ← "Test"
OPEN MyFile FOR WRITE
    WRITEFILE, Text
CLOSEFILE(MyFile)
```

ii

```
OPEN MyFile FOR READ
    READFILE, Text
    OUTPUT Text
CLOSEFILE(MyFile)
```

c

Python

```
MyFile = open("MyFile.txt","w")
Text = "Test"
MyFile.write(Text)

print("The file contains this line of text")
MyFile = open("MyFile.txt","r")
Text = MyFile.read()
print(Text)
```

Visual Basic

```
Imports System.IO
Module Module1
    Sub Main()
        Dim Text As String
        Dim objMyFileWrite As StreamWriter
        Dim objMyFileRead As StreamReader
        objMyFileWrite = New StreamWriter("MyFile.txt")
        Console.Write("Please enter a line of text  ")
        Text = "Test"
        objMyFileWrite.WriteLine(Text)
        objMyFileWrite.Close()

        objMyFileRead = New StreamReader("MyFile.txt")
        Text = objMyFileRead.ReadLine()
        Console.Write("The file contains this line of text ")
        Console.WriteLine(Text)
        objMyFileRead.Close()
    End Sub
End Module
```

Java

```
import java.io.BufferedReader;
import java.io.PrintWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

class TextFile {

    public static void main(String[] args) {
        String Text;
        try {
            FileWriter myFileWriter = new FileWriter("MyFile.txt", false);
            PrintWriter myPrintWriter = new PrintWriter(myFileWriter);

            Text = "Test";
            myPrintWriter.printf("%s" + "%n", Text);
            myPrintWriter.close();
      } catch (IOException e) {
          e.printStackTrace();
      }

      try {
        FileReader myFileReader  = new FileReader("MyFile.txt");
        BufferedReader myBufferReader  = new BufferedReader(myFileReader);

        Text = myBufferReader.readLine();
        System.out.print("The file contains this line of text ")
        System.out.println(Text);

        myFileReader.close();

      } catch (IOException e) {
          e.printStackTrace();
      }
   }

}
```

14 a–e:

Python

```
#### Part b ###
def NewPassword():
    MyPassword = input("Please enter your new password ")
    Valid = True
    Length = len(MyPassword)
    if Length > 20:
        print("Too long")
        Valid = False
    if Length < 10:
        print("Too short")
        Valid = False
```

```
Counter = 0
 NoSpace = True
 while (Counter < Length and NoSpace):
     if MyPassword[Counter:Counter + 1] == " ":
         print("Spaces not allowed")
         Valid = False
         NoSpace = False
     Counter = Counter + 1

Counter = 0
 UpperCase = False
 while (Counter < Length and not UpperCase):
     if "A" <= MyPassword[Counter:Counter + 1] <= "Z":
         UpperCase = True
     Counter = Counter + 1
 if (not UpperCase):
     print("No uppercase letter")
     Valid = False

 Counter = 0
 Digit = False
 while (Counter < Length and not Digit):
     if "0" <= MyPassword[Counter:Counter + 1] <= "9":
         Digit = True
     Counter = Counter + 1
 if (not Digit):
     print("No digit")
     Valid = False

if Valid:
     print("Password accepted")
     MyFile = open("MyPassword.txt","w")    #### Part c ###
     MyFile.write(MyPassword)               #### Part c ###
 else:
     print("Password rejected")
```

```
#### Part d ###

def CheckPassword():
    MyPassword = input("Please enter your password ")
    MyFile = open("MyPassword.txt","r")
    StoredPassword = MyFile.read()
    if MyPassword == StoredPassword:
        print("Password matches")
    else:
        print("Password does not match")


#### Part e ###

def ChangePassword():
    MyPassword = input("Please enter your password ")
    MyFile = open("MyPassword.txt","r")
    StoredPassword = MyFile.read()

    if MyPassword == StoredPassword:
        NewPassword()
    else:
        print("Password incorrect cannot change")

#### Part a ###

Quit = False
while not Quit:
    print ("    Password Management")
    print (" 1.  Enter a new password")
    print (" 2.  Check your password")
    print (" 3.  Change your password")
    print (" 4.  Quit")

    option = int(input("Please enter your chosen option "))
```

```
    if option == 1:
        NewPassword()
    elif option == 2:
        CheckPassword()
    elif option == 3:
        ChangePassword()
    elif option == 4:
        Quit = True
    else:
        print ("Incorrect option chosen")
```

Visual Basic

```
Imports System.IO
Module Module1

'*** Part a ***
    Sub Main()
        Dim Opt As Integer
        Dim Quit As Boolean = False
        Dim MyPassword As String

        While Not Quit
            Console.WriteLine("    Password Management")
            Console.WriteLine(" 1. Enter a new password")
            Console.WriteLine(" 2. Check your password")
            Console.WriteLine(" 3. Change your password")
            Console.WriteLine(" 4. Quit")

            Console.Write("Please enter your chosen option ")
            Opt = Integer.Parse(Console.ReadLine())
```

```
            Select Case Opt
                Case 1
                    Console.Write("Please enter your password ")
                    MyPassword = Console.ReadLine()
                    NewPassword(MyPassword)
                Case 2
                    Console.Write("Please enter your password ")
                    MyPassword = Console.ReadLine()
                    CheckPassword(MyPassword)
                Case 3
                    Console.Write("Please enter your password ")
                    MyPassword = Console.ReadLine()
                    ChangePassword(MyPassword)
                Case 4
                    Quit = True
                Case Else
                    Console.WriteLine("Incorrect option chosen")
            End Select

            Console.ReadKey()

        End While

    End Sub
'******
```

```
'*** Part b ***
    Sub NewPassword(MyString As String)
        Dim Length, Counter As Integer
        Dim Valid, NoSpace, UpperCase, Digit As Boolean
        Dim objMyFileWrite As StreamWriter

        Valid = True
        Length = Len(MyString)
        If Length > 20 Then
            Console.WriteLine("Too long")
            Valid = False
        End If
        If Length < 10 Then
            Console.WriteLine("Too short")
            Valid = False
        End If

        Counter = 0
        NoSpace = True
        While Counter < Length And NoSpace
            If MyString.Substring(Counter, 1) = " " Then
                Console.WriteLine("Spaces not allowed ")
                Valid = False
                NoSpace = False
            End If
            Counter = Counter + 1
        End While
```

```
Counter = 0
UpperCase = False
While Counter < Length And Not UpperCase
    If MyString.Substring(Counter, 1) >= "A" And
    MyString.Substring(Counter, 1) <= "Z" Then
        UpperCase = True
    End If
    Counter = Counter + 1
End While
If Not UpperCase Then
    Console.WriteLine("No uppercase letter")
    Valid = False
End If

Counter = 0
Digit = False
While Counter < Length And Not Digit
    If MyString.Substring(Counter, 1) >= "0" And
    MyString.Substring(Counter, 1) <= "9" Then
        Digit = True
    End If
    Counter = Counter + 1
End While
If Not Digit Then
    Console.WriteLine("No digit")
    Valid = False
End If
If Valid Then
    Console.WriteLine("Password accepted. Press any
    key to continue.")
    objMyFileWrite = New
    StreamWriter("MyPassword.txt")'*** Part c ***
    objMyFileWrite.WriteLine(MyString)'*** Part c ***
    objMyFileWrite.Close()'*** Part c ***
Else
```

```
            Console.WriteLine("Password rejected. Press any
            key to continue.")
        End If

    End Sub
'******

'**** Part d ***
    Sub CheckPassword(MyString As String)
        Dim objMyFileRead As StreamReader
        Dim StoredPassword As String
        objMyFileRead = New StreamReader("MyPassword.txt")
        StoredPassword = objMyFileRead.ReadLine
        objMyFileRead.Close()

        If MyString = StoredPassword Then
            Console.Write("Password matches. Press any key to
            continue.")
        Else
            Console.WriteLine("Password does not match. Press
            any key to continue.")
        End If

    End Sub
'******

'*** Part e ***
    Sub ChangePassword(MyString As String)
        Dim NewWord As String
        Dim objMyFileRead As StreamReader
        Dim StoredPassword As String
        objMyFileRead = New StreamReader("MyPassword.txt")
        StoredPassword = objMyFileRead.ReadLine()
        objMyFileRead.Close()
```

```
        If MyString = StoredPassword Then
            Console.Write("Please enter your new password ")
            NewWord = Console.ReadLine()
            NewPassword(NewWord)
        Else
            Console.WriteLine("Password incorrect cannot
            change. Press any key to continue.")
        End If

    End Sub
'******

End Module
```

Java

```
import java.util.*;
import java.io.BufferedReader;
import java.io.PrintWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

class Chapter8Q14 {
 // *** Part b ***
    static void NewPassword(String MyString) {
        int Length;
        Length = MyString.length();
        boolean Valid = true;

        if (Length > 20) {
            System.out.println("Too long");
            Valid = false;
        }
```

```
 if (Length < 10) {
     System.out.println("Too short");
     Valid = false;
 }

int Counter = 0;
boolean NoSpace = true;

while ((Counter < Length) & NoSpace) {
  if (MyString.substring(Counter,Counter +
  1).compareTo( " ") == 0) {
      System.out.println("Spaces not allowed");
      Valid = false;
      NoSpace = false;
  }
  Counter ++;

  }

Counter = 0;
boolean UpperCase = false;
while ((Counter < Length) & !UpperCase) {
   if (MyString.charAt(Counter) >= 'A' &&
   MyString.charAt(Counter) <= 'Z') {
       UpperCase = true;
   }
   Counter ++;
}
if(!UpperCase) {
    System.out.println("No uppercase letter");
    Valid = false;
}
```

```
        Counter = 0;
        boolean Digit = false;
        while ((Counter < Length) & !Digit) {

           if (MyString.charAt(Counter) >= '0' &&
           MyString.charAt(Counter) <= '9') {
           Digit = true;
           }
           Counter ++;
        }
        if(!Digit) {
            System.out.println("No digit");
            Valid = false;
        }



        if (Valid) {
            System.out.println("Password accepted");
            try {
                 FileWriter myFileWriter = new
                 FileWriter("MyPassword.txt", false); //***
                 Part c ***
                 PrintWriter myPrintWriter = new
                 PrintWriter(myFileWriter); //*** Part c ***
                 myPrintWriter.printf("%s" + "%n", MyString);
                 //*** Part c ***
                 myPrintWriter.close(); //*** Part c ***
            } catch (IOException e) { //*** Part c ***
                 e.printStackTrace(); //*** Part c ***
            }
}
   else {
        System.out.println("Password rejected");
    }
}
```

```
// ******

// *** Part d ***
   static void CheckPassword(String MyString) {
      String StoredPassword = "";

      try {
            FileReader myFileReader  = new
            FileReader("MyPassword.txt");
            BufferedReader myBufferReader  = new
            BufferedReader(myFileReader);
            StoredPassword = myBufferReader.readLine();
            myFileReader.close();
      } catch (IOException e) {
            e.printStackTrace();
      }

      if (MyString.compareTo(StoredPassword) == 0) {
            System.out.println("Password matches");
      }
      else {
            System.out.println("Password does not match");
      }

   }
//******
```

```
// *** Part e ***
   static void ChangePassword(String MyString) {
      Scanner myObj = new Scanner(System.in);
      String StoredPassword = "";
      String NewWord = "";

      try {
            FileReader myFileReader  = new
            FileReader("MyPassword.txt");
            BufferedReader myBufferReader  = new
            BufferedReader(myFileReader);
            StoredPassword = myBufferReader.readLine();
            myFileReader.close();
      } catch (IOException e) {
            e.printStackTrace();
      }

      if (MyString.compareTo(StoredPassword) == 0) {
            System.out.print("Please enter your new password ");
            NewWord = myObj.nextLine();
            NewPassword(NewWord);
      }
      else {
            System.out.println("Password incorrect cannot
            change");
      }
   }
//******
```

```
// *** Part a ***
   public static void main(String args[]) {
       Scanner myObj = new Scanner(System.in);
       boolean Quit = false;
       while (!Quit) {
            System.out.println("     Password Management");
            System.out.println(" 1   Enter a new password");
            System.out.println(" 2   Check your password");
            System.out.println(" 3   Change your password");
            System.out.println(" 4   Quit");

            System.out.print("Please enter your chosen option ");
            int option = myObj.nextInt();

            String MyPassword = myObj.nextLine();



           switch (option) {
               case 1:
                    System.out.print("Please enter your
                    password ");
                    MyPassword = myObj.nextLine();
                    NewPassword(MyPassword);
                    break;
               case 2:
                    System.out.print("Please enter your
                    password ");
                    MyPassword = myObj.nextLine();
                    CheckPassword(MyPassword);
                    break;
               case 3:
                    System.out.print("Please enter your
                    password ");
                    MyPassword = myObj.nextLine();
                    ChangePassword(MyPassword);
                    break;
               case 4:
                    Quit = true;
                    break;
```

```
                    default:
                        System.out.println("Incorrect option
                        chosen");
                }
            }
        //******
        }
    }
```

b

Test data	Expected output	Actual output
Password	Too short No digit Password rejected	
Password99	Password accepted	
password99	No uppercase letter Password rejected	
Password99!	Password accepted	
ExtraLargePassword9999999	Too long Password rejected	
2 1	Too short Spaces not allowed No uppercase letter Password rejected	

15 Important points to note when writing your program:

- There is not one 'correct' solution to this problem. Any solution is acceptable but your program must:
  - perform all the tasks set out in the question
  - use a range of appropriate programming techniques that cover all the tasks set out in the question. These tasks must be executed by your program in a logical order
  - use appropriate data structures, which have been given meaningful names, to store all the data mentioned in the question. If any data structures are named in the question, then those names must be used in your program
  - be fully commented to ensure that it can be understood
- The array name must be `Game[]`
- All other data structures must have meaningful names.

- Procedures must be used, for example to:
  - set up an empty array at the start of each game
  - display the contents of the array
  - enter a player's move – this could use a parameter for `O` or `X`
  - check for a winner
- Your program must be fully commented.
- Your answer to 12 will help you to start your program.

9 Databases

1 a
- table: a collection of records …
- … where each record contains data about a similar item, e.g. a student.
- record: a collection of fields about the same item …
- … there are the same number of fields in each record.
- field: a single piece of data …
- … that is given a data type.

b Four from:
- text/alphanumeric – e.g. a description of a part
- character – e.g. S, M or L for the size of a part
- Boolean – e.g. true or false, sold or not
- integer – e.g. number of parts in stock
- real – e.g. price of an item 12.99
- date/time – e.g. a date such as 18/12/2020

2 a

Field 1: Type
Data type: text
Reason: as the name given to the type of ice cream is a series of characters
Sample: choc ice

Field 2: Flavour
Data type: text
Reason: as the name given to the flavour of ice cream is a series of characters
Sample: vanilla

Field 3: Size
Data type: text
Reason: as the size of the ice cream is stored as a word
Sample: Small

Field 4: NumberInStock
Data type: integer
Reason: as the number in stock will be a whole number
Sample: 56

Field 5: ReOrderLevel

Data type: integer

Reason: as the re-order level will be a whole number

Sample: 24

b i A primary key is required to uniquely identify each record.

ii All of the fields could be repeated in another record.

iii Name: IceCreamID

Date type: text

Reason: so the field can contain letters and digits

Sample: IC007

c

ICECREAM

Field Name	Data Type	
IDIceCream	Short Text	8 characters only
Type	Short Text	
Flavour	Short Text	
Size	Short Text	
NumberInStock	Number	
ReOrderLevel	Number	

d

ICECREAM

IDIceCream	Type	Flavour	Size	NumberInStock	ReOrderLevel
IC000001	Lolly	Strawberry	Large	35	20
IC000002	Lolly	Raspberry	Small	14	20
IC000003	Choc Ice	Vanilla	Medium	60	30
IC000004	Choc Ice	Chocolate	Medium	30	20
IC000005	Tub	Vanilla	Small	0	10
IC000006	Tub	Vanilla	Large	15	10

3 a Structured Query Language (SQL) is the standard query language …

… for writing scripts to obtain information from a database.

b i
- SELECT: Fetches specified fields (columns) from a table …
- … queries always begin with `SELECT`

ii
- FROM: Identifies the table to use …
- … queries always include `FROM`.

iii
- WHERE: Includes only records (rows) in a query that match a given condition …
- … `WHERE` is an optional command.

iv
- SUM: Adds the values in a specified field …
- … must be an integer or real field.

c `ORDER BY` and `COUNT`

d **Query 1**

```
SELECT TYPE, Size
FROM ICECREAM
WHERE ((NumberInStock)>0)
ORDER BY NameIceCream;
```

Query 2

```
SELECT SUM(NumberInStock)
FROM ICECREAM;
```

Query 3

```
SELECT Count(NumberInStock)
FROM ICECREAM
WHERE(NumberInStock < ReOrderLevel);
```

4 **a** **i**
- Licence …
- … as it uniquely identifies the teacher.

ii

Name:	text
Title:	text
Licence:	text
Gender:	char or Boolean
Subject:	text
Class:	text

iii Learner's own database

b **i**
- Name …
- … so that any copying errors during data entry can be identified.

ii
- Any of Gender, Title, Licence, Subject, Class could be validated …
- … as there are only a small number of different entries allowed for each.

iii Learner's own validation

c **i** Miss Sing

Mr Ling

ii
```
SELECT Title, Name
  FROM TEACHER
  WHERE Subject = "Science"
  ORDER BY Name;
```

iii
```
SELECT Title, Name
  FROM TEACHER
  WHERE (Subject = "Science" OR Subject = "Mathematics")
  ORDER BY Name;
```

iv
```
SELECT COUNT (Subject)
  FROM TEACHER
  WHERE (Subject = "Mathematics");
```

v Learner's own check

d
```
SELECT Name, Class
  FROM TEACHER
  ORDER BY Class;
```

10 Boolean logic

1 **a** OR gate

b NAND gate

c XOR gate

2

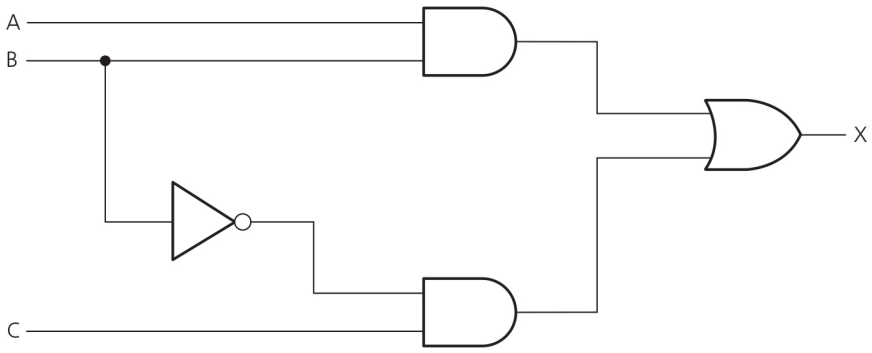
A	B	C	Working area	X
0	0	0		**0**
0	0	1		**0**
0	1	0		**0**
0	1	1		**1**
1	0	0		**1**
1	0	1		**1**
1	1	0		**0**
1	1	1		**1**

3 **a** `(A AND B) OR (NOT B AND C)`

or

`(A.B) + (`$\overline{B}$`.C)`

b



4 **a**

A	B	Working area	X
0	0		**0**
0	1		**1**
1	0		**1**
1	1		**1**

b OR gate

5 a

A	B	C	Working area	X
0	0	0		**0**
0	0	1		**1**
0	1	0		**0**
0	1	1		**1**
1	0	0		**0**
1	0	1		**1**
1	1	0		**0**
1	1	1		**1**

b As the output X exactly matches input C, the entire logic circuit could be removed and the output connected directly to the input C.

6 a

A	B	C	Working area	X
0	0	0		**1**
0	0	1		**1**
0	1	0		**0**
0	1	1		**0**
1	0	0		**1**
1	0	1		**1**
1	1	0		**0**
1	1	1		**1**

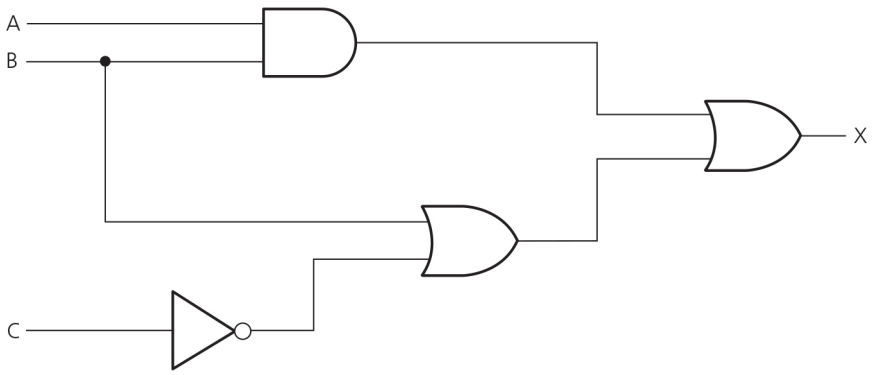
b AND gate

7 a `(A AND NOT B) AND (B OR C)`

or

$(A.\overline{B}).(B + C)$

b

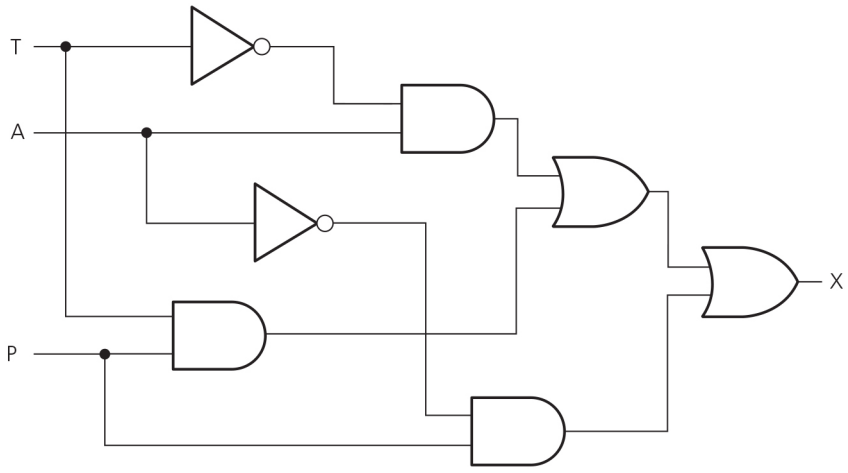


8 a ((NOT T AND A) OR (T AND P)) OR (NOT A AND P)

or

$(\overline{T}.A + T.P) + (\overline{A}.P)$

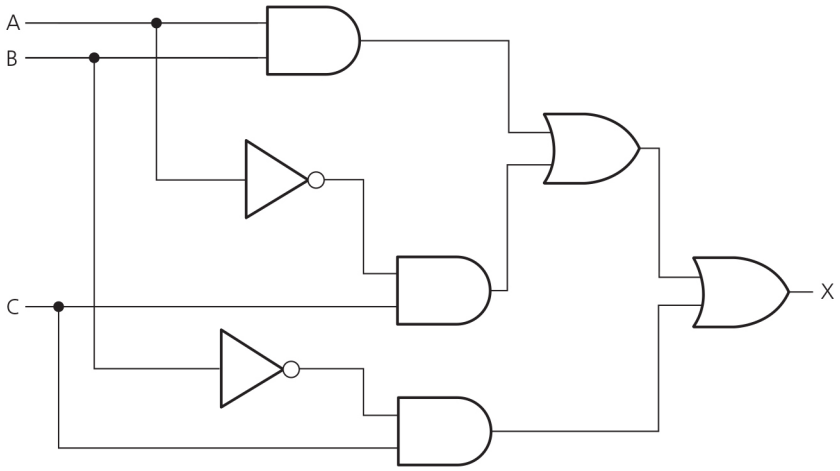
b



c

T	A	P	Working area	X
0	0	0		**0**
0	0	1		**1**
0	1	0		**1**
0	1	1		**1**
1	0	0		**0**
1	0	1		**1**
1	1	0		**0**
1	1	1		**1**

9 a



b

A	B	C	Working area	X
0	0	0		**0**
0	0	1		**1**
0	1	0		**0**
0	1	1		**1**
1	0	0		**0**
1	0	1		**1**
1	1	0		**1**
1	1	1		**1**

c

X	Y	Z	Working area	Q
0	0	0		**0**
0	0	1		**0**
0	1	0		**0**
0	1	1		**1**
1	0	0		**0**
1	0	1		**0**
1	1	0		**1**
1	1	1		**1**

10 a `((A AND B) AND NOT A) OR (B OR NOT C)`

or

$(A.B).\overline{A} + (B + \overline{C})$

A	B	C	Working area	X
0	0	0		**1**
0	0	1		**0**
0	1	0		**1**
0	1	1		**1**
1	0	0		**1**
1	0	1		**0**
1	1	0		**1**
1	1	1		**1**

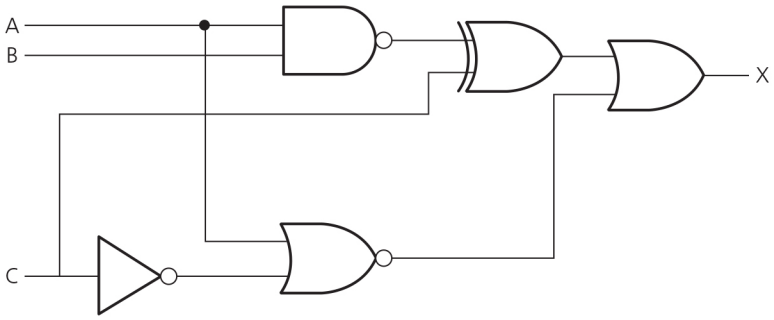
11 a `(A AND B) OR (NOT A AND NOT B)`

or

$(A.B) + (\overline{A}.\overline{B})$

b

12 a



b

A	B	C	Working area	X
0	0	0		**1**
0	0	1		**1**
0	1	0		**1**
0	1	1		**1**
1	0	0		**1**
1	0	1		**0**
1	1	0		**0**
1	1	1		**1**