

IMRAN KHAN

## 2.1 Algorithm design and problem-solving

Candidates should be able to:

### 2.1.1 Problem-solving and design

- show understanding that every computer system is made up of sub-systems, which in turn are made up of further sub-systems
- use top-down design, structure diagrams, flowcharts, pseudocode, library routines and subroutines
- work out the purpose of a given algorithm
- explain standard methods of solution
- suggest and apply suitable test data
- understand the need for validation and verification checks to be made on input data (validation could include range checks, length checks, type checks and check digits)
- use trace tables to find the value of variables at each step in an algorithm
- identify errors in given algorithms and suggest ways of removing these errors
- produce an algorithm for a given problem (either in the form of pseudocode or flowchart)
- comment on the effectiveness of a given solution

### 2.1.2 Pseudocode

- understand and use pseudocode for assignment, using ←
  - understand and use pseudocode, using the following conditional statements:  
IF ... THEN ... ELSE ... ENDIF  
CASE ... OF ... OTHERWISE ... ENDCASE
  - understand and use pseudocode, using the following loop structures:  
FOR ... TO ... NEXT  
REPEAT ... UNTIL  
WHILE ... DO ... ENDWHILE
  - understand and use pseudocode, using the following commands and statements:  
INPUT and OUTPUT (e.g. READ and PRINT)  
totalling (e.g. Sum ← Sum + Number)  
counting (e.g. Count ← Count + 1)
- (Candidates are advised to try out solutions to a variety of different problems on a computer using a language of their choice; no particular programming language will be assumed in this syllabus.)

## Flow Charts

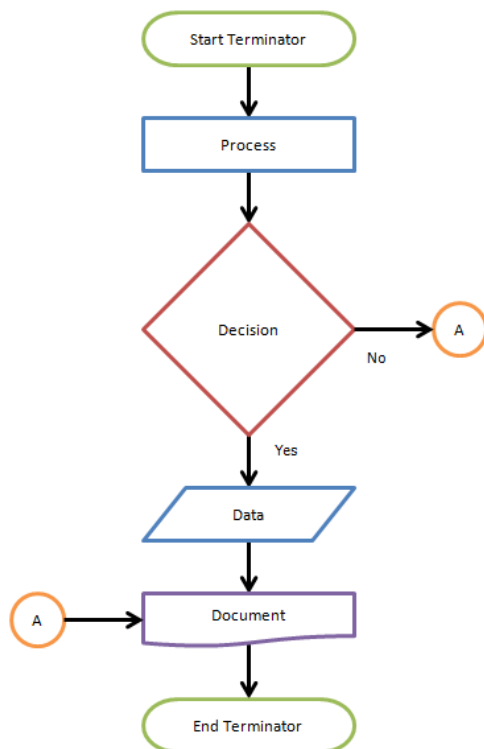
A **flowchart** is a type of diagram that represents an algorithm, workflow or process, showing the steps as boxes of various kinds, and their order by connecting them with arrows.

### Common Flowchart Symbols

Different flow chart symbols have different meanings. The most common flow chart symbols are:

- **Terminator:** An oval flow chart shape indicating the start or end of the process.
- **Process:** A rectangular flow chart shape indicating a normal process flow step.
- **Decision:** A diamond flow chart shape indicating a branch in the process flow.
- **Connector:** A small, labeled, circular flow chart shape used to indicate a jump in the process flow. (Shown as the circle with the letter "A", below.)
- **Data:** A parallelogram that indicates data input or output (I/O) for a process.
- **Document:** Used to indicate a document or report (see image in sample flow chart below).

A simple flow chart showing the symbols described above can be seen below:



## Pseudocode

### Common pseudocode terms

#### 1.1 counting

Counting in 1s is quite simple; use of the statement  $\text{count} = \text{count} + 1$  will enable counting to be done (e.g. in controlling a repeat loop). The statement literally means: **the (new) count = the (old) count + 1**

It is possible to count in any increments just by altering the numerical value in the statement (e.g.  $\text{count} = \text{count} - 1$  counts backwards)

#### 1.2 totalling

To add up a series numbers the following type of statement should be used:

$\text{total} = \text{total} + \text{number}$

This literally means **(new) total = (old) total + value of number**

#### 1.3 input/output

Input and output indicated by the use of the terms input number, output total, print total, print "result is" x and so on.

#### 1.4 branching

There are two common ways of branching:

**case of ..... otherwise ..... endcase**

**if ..... then ..... else ..... endif**

#### 1.5 Looping

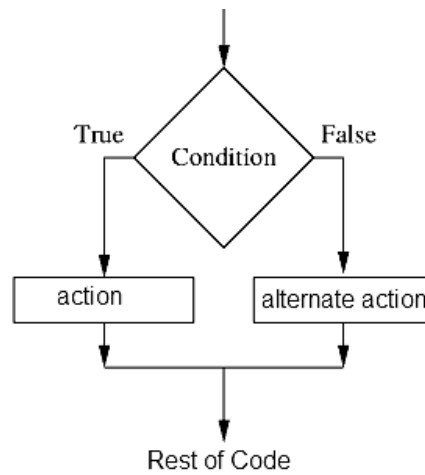
There are three terms of looping

- FOR.. TO ..NEXT
- REPEAT .....UNTIL
- WHILE ... DO .....ENDWHILE

## Selection Structure

A high-level programming language statement that compares two or more sets of data and tests the results. If the results are true, the THEN instructions are taken; if not, the ELSE instructions are taken. The following is a BASIC example:

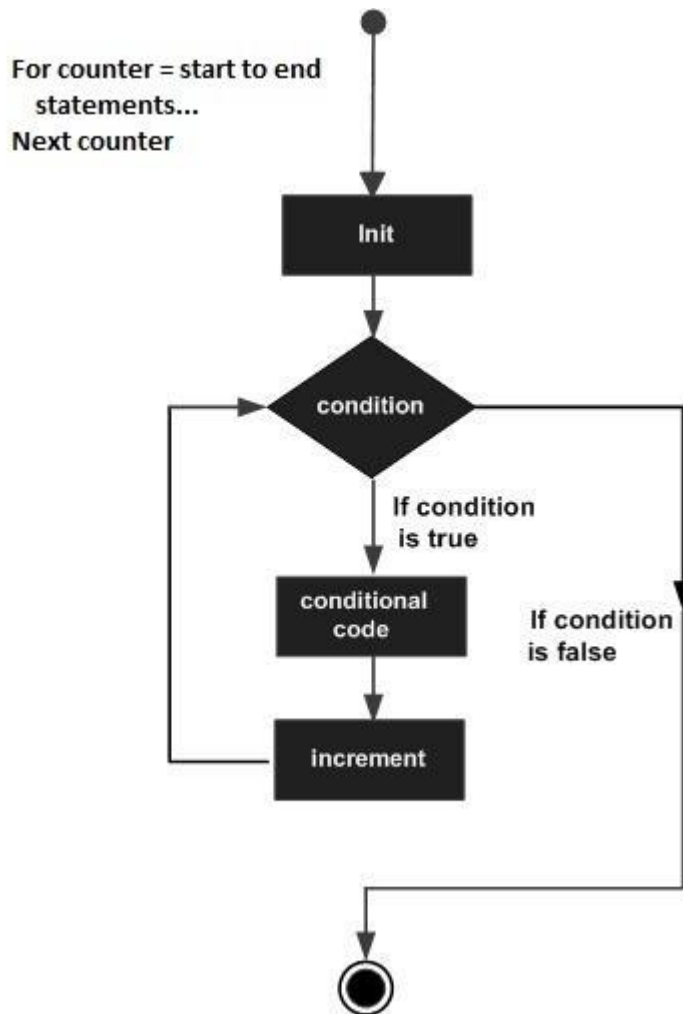
```
if answer = "y" then print "Yes"  
else print "No"
```



```
If [your condition here]  
    Your code here  
Else  
    Your code Here  
End If
```

### For..... Next Loop:

For loop is used to repeat a single programming instruction or multiple programming instructions for number of times depending upon the initial and final values in the condition.



## While... End While Loop

It executes a series of statements as long as a given condition is True.

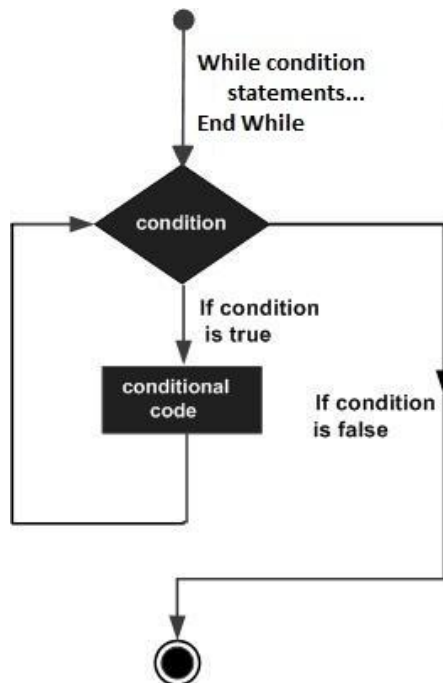
The syntax for this loop construct is:

```
While condition
  [ statements ]
  [ Continue While ]
  [ statements ]
  [ Exit While ]
  [ statements ]
End While
```

Here, statement(s) may be a single statement or a block of statements. The condition may be any expression, and true is logical true. The loop iterates while the condition is true.

When the condition becomes false, program control passes to the line immediately following the loop.

### Flow Diagram:



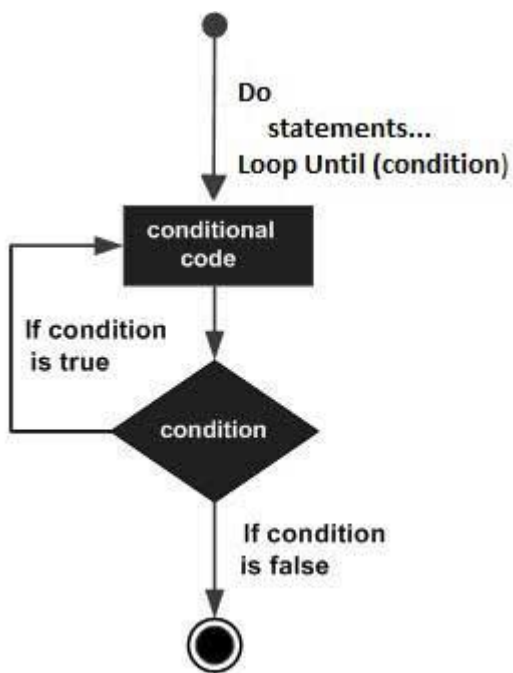
Here, key point of the *While* loop is that the loop might not ever run. When the condition is tested and the result is false, the loop body will be skipped and the first statement after the while loop will be executed.

## REPEAT...UNTIL

REPEAT...UNTIL loops are used to repetitively execute a subject statement until a condition is true. The condition is checked after the subject statement is executed. Therefore, the subject statement is always executed at least once. See [Definition of True and False](#) for details on how the “truth” of an expression is determined.

### Syntax

- REPEAT *statement* UNTIL *expression*
- or
- REPEAT BEGIN
- *statements*
- ENDREP UNTIL *expression*





### Writing algorithms using pseudocode

The following five examples use the above pseudocode terms. These are the same problems discussed in section 3.1 using flow charts – both methods are acceptable ways of representing an algorithm.

#### Example 1

A town contains 5000 houses. Each house owner must pay tax based on the value of the house. Houses over \$200 000 pay 2% of their value in tax, houses over \$100 000 pay 1.5% of their value in tax and houses over \$50 000 pay 1% of their value in tax. All others pay no tax. Write an algorithm to solve the problem using pseudocode.

```
for count = 1 to 5000
input house
if house > 50 000 then tax = house * 0.01
else if house > 100 000 then tax = house * 0.015
else if house > 200 000 then tax = house * 0.02
else tax = 0
print tax
next
```

Notes: (1) a **while** loop or a **repeat** loop would have worked just as well  
(2) the use of **endif** isn't essential in the pseudocode  
For example,

```
count = 0
while count < 5001
input house
if house > 50 000 then tax = house * 0.01
else if house > 100 000 then tax = house * 0.015
else if house > 200 000 then tax = house * 0.02
else tax = 0
endif
endif
endif
print tax
count = count + 1
endwhile
```

**EXERCISE:** Re-write the above algorithm using a **repeat** loop and modify the **if ...**

**then ... else** statements to include both parts of the house price range.  
(e.g. **if** house > 50000 and house <= 100000 **then** tax = house \* 0.01)

### Example 3

Write an algorithm using pseudocode which takes temperatures input over a 100 day period (once per day) and output the number of days when the temperature was below 20C and the number of days when the temperature was 20C or above.  
(NOTE: since the number of inputs is known, a **for ... to** loop can be used. However, a **while** loop or a **repeat** loop would work just as well).

```
total1 = 0: total2 = 0
for days = 1 to 100
  input temperature
  if temperature < 20 then total1 = total1 + 1
  else total2 = total2 + 1
  endif
next
print total1, total2
```

This is a good example of an algorithm that could be written using the **case** construct rather than **if ... then ... else**.

The following section of code replaces the statements *if temperature < 20 then .....*  
**endif**:

```
case temperature of
  1: total1 = total1 + 1
  2: total2 = total2 + 1
endcase
```

### Example 4

Write an algorithm using pseudocode which:

- inputs the top speeds of 5000 cars
- outputs the fastest speed and the slowest speed
- outputs the average speed of all the 5000 cars

(NOTE: Again since the actual number of data items to be input is known any one of the three loop structures could be used. It is necessary to set values for the fastest (usually set at zero) and the slowest (usually set at an unusually high value) so that each input can be compared. Every time a value is input which > the value stored in fastest then this input value replaces the existing value in fastest; and similarly for slowest).

```
fastest = 0: count = 0
slowest = 1000
repeat
input top_speed
total = total + top_speed
if top_speed > fastest then fastest = top_speed
if top_speed < slowest then slowest = top_speed
endif
endif
count = count + 1
until count = 5000
average = total * 100/5000
print fastest, slowest, average
```

### Example 5

A shop sells books, maps and magazines. Each item is identified by a unique 4 – digit code. All books have a code starting with a 1, all maps have a code starting with a 2 and all magazines have a code beginning with a 3. The code 9999 is used to end the program.

Write an algorithm using pseudocode which input the codes for all items in stock and outputs the number of books, maps and magazine in stock.

Include any validation checks necessary.

(NOTE: A 4-digit code implies all books have a code lying between 1000 and 1999, all maps have a code lying between 2000 and 2999 and all magazines a code lying between 3000 and 3999. Anything outside this range is an error)

```
books = 0: maps = 0: mags = 0
repeat
input code
if code > 999 and code < 2000 then books = books + 1
else if code > 1999 and code < 3000 then maps = maps + 1
else if code > 2999 and code < 4000 then mags = mags + 1
else print "error in input"
endif:endif:endif
until code = 9999
print books, maps, mags
```

IMRAN KHAN

## 2.2 Programming

Candidates should be able to:

### 2.2.1 Programming concepts

- declare and use variables and constants
- understand and use basic data types: Integer, Real, Char, String and Boolean
- understand and use the concepts of sequence, selection, repetition, totalling and counting
- use predefined procedures/functions

### 2.2.2 Data structures; arrays

- declare the size of one-dimensional arrays, for example: A[1:n]
- show understanding of the use of a variable as an index in an array
- read values into an array using a FOR ... TO ... NEXT loop

## VB Programming

### Fundamentals of Programming: A program

Example will print two values and their sum.

```
Module Example
  Sub Main()
    Dim value1 As Integer
    Dim value2 As Integer
    Dim sum As Integer
    value1 = 1
    value2 = 2
    sum = value1 + value2
    '
    ' or :
    ' Dim value1 As Integer = 1
    ' Dim value2 As Integer = 2
    ' Dim sum As Integer = value1 + value2
    '
    ' or :
    ' Dim value1 As Integer = 1, value2 As Integer = 2, _
    '     sum As Integer = value1 + value2
    '
    Console.WriteLine(value1 & " + " & value2 & _
                      " = " & sum)
    '
    Console.ReadLine()
  End Sub
End Module
```

Output :

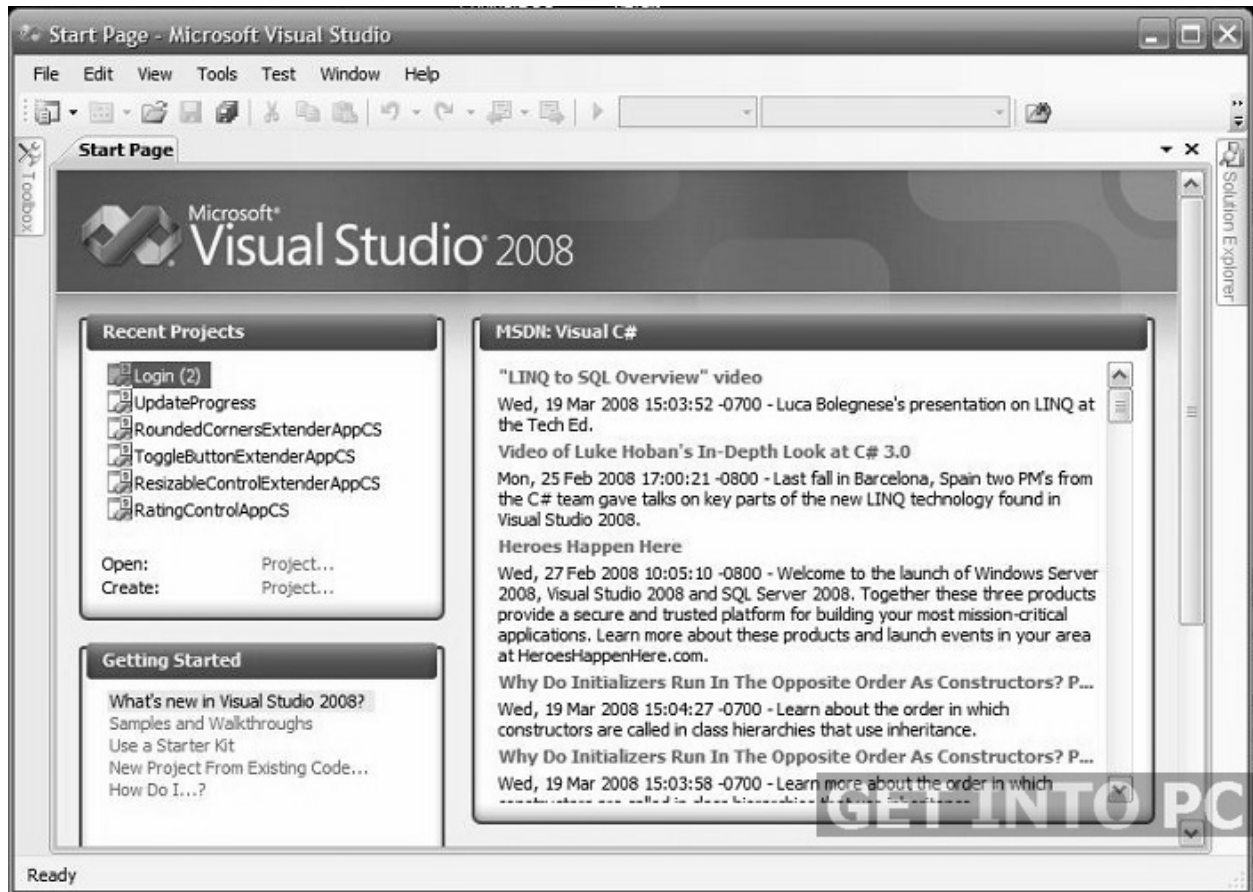
```
1 + 2 = 3
```

### Arithmetical Operators

The above example illustrates the use of the addition operator +.

Other arithmetical operators are :

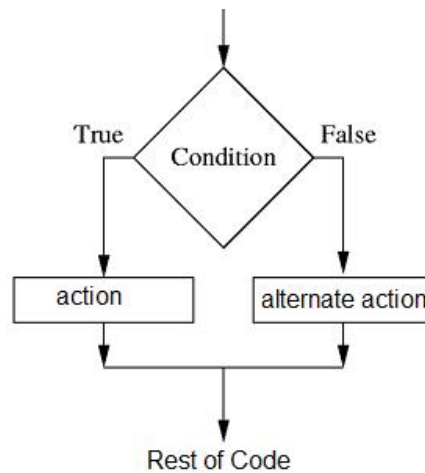
- subtraction operator -
- multiplication operator \*
- division operator /
- integral division operator \
- modulo operator Mod
- exponential operator ^



## Selection Structure

A high-level programming language statement that compares two or more sets of data and tests the results. If the results are true, the THEN instructions are taken; if not, the ELSE instructions are taken. The following is a BASIC example:

```
if answer = "y" then print "Yes"  
else print "No"
```



```
If [your condition here]  
    Your code here  
Else  
    Your code Here  
End If
```

We also could place 3 selections ( Ifs ) in a sequential order.

```
Module Example2  
    Sub Main()  
        Console.WriteLine("Value ?")  
        Dim value As Integer = Console.ReadLine()  
        '  
        If value = 0 Then Console.WriteLine("Zero.")  
        If value > 0 Then Console.WriteLine("Positive value.")  
        If value < 0 Then Console.WriteLine("Negative value.")  
        '  
        Console.ReadLine()  
    End Sub  
End Module
```

Output :

```
Value ?  
0  
Zero.
```



Another variation can be build using ElseIf.

```
Module Example3
  Sub Main()
    Console.WriteLine("Value ?")
    Dim value As Integer = Console.ReadLine()
    '
    If value = 0 Then
      Console.WriteLine("Zero.")
    ElseIf value > 0 Then
      Console.WriteLine("Positive value.")
    Else
      Console.WriteLine("Negative value.")
    End If
    '
    Console.ReadLine()
  End Sub
End Module
```

Output :

```
Value ?
0
Zero.
```

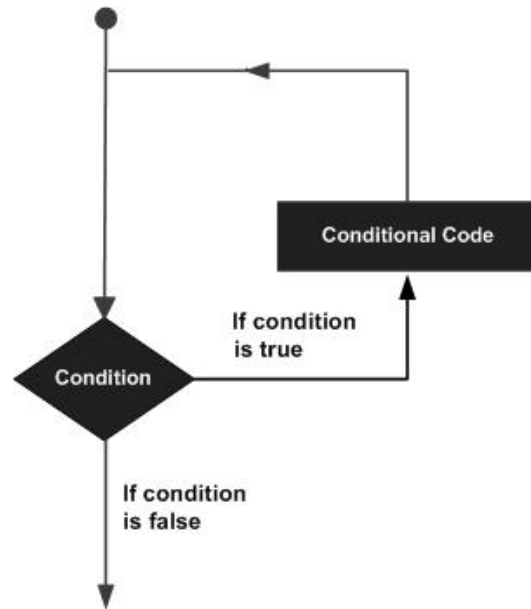
## MAKING A MARKSSHEET ON THE BASIS OF TEST MARKS

Module Module1

```
Sub Main()  
    For count As Integer = 1 To 5  
  
        'declaration of variable for test 1  
        Dim test1 As Single  
        'declaration of variable for test 2  
        Dim test2 As Single  
        'declaration of variable for test 3  
        Dim test3 As Single  
        'declaration of variable for total  
        Dim total As Single  
        'declaration of variable for average  
        Dim avg As Single  
        'declaration of variable for percentage  
        Dim perc As Single  
        'input for test1  
        Console.WriteLine("marks for test 1 out of 20")  
        test1 = Console.ReadLine  
        'input for test2  
        Console.WriteLine("marks for test 2 out of 25")  
        test2 = Console.ReadLine  
        'input for test3  
        Console.WriteLine("marks for test 3 out of 35")  
        test3 = Console.ReadLine  
        'input, calculate and output total, average and percentage  
        total = test1 + test2 + test3  
        avg = (test1 + test2 + test3) / 3  
        perc = ((test1 + test2 + test3) / 80) * 100  
        Console.WriteLine("Total of marks is: {0}", total)  
        Console.WriteLine("average of marks is: {0}", avg)  
        Console.WriteLine("percentage of marks is: {0}", perc)  
        'grading  
        If perc >= 90 Then  
            Console.WriteLine("A*")  
        ElseIf perc >= 80 Then  
            Console.WriteLine("A")  
        ElseIf perc >= 70 Then  
            Console.WriteLine("B")  
        ElseIf perc >= 60 Then  
            Console.WriteLine("C")  
        ElseIf perc >= 50 Then  
            Console.WriteLine("D")  
        Else  
            Console.WriteLine("fail")  
        End If  
    Next  
    Console.ReadKey()  
  
End Sub  
  
End Module
```

## Looping Structure:

A loop statement allows us to execute a statement or group of statements multiple times.



VB.Net provides following types of loops to handle looping requirements.

1. For..... Next Loop
2. Do...Loop
3. While... End While Loop

### **For..... Next Loop:**

For loop is used to repeat a single programming instruction or multiple programming instructions for number of times depending upon the initial and final values in the condition.

### **Why use Iterations**

Following example gives us all numbers from one to three.

```
Module Module1
    Sub Main()
        Console.WriteLine(1)
        Console.WriteLine(2)
        Console.WriteLine(3)
        '
        Console.ReadLine()
    End Sub
End Module
```

## Output:

```
1
2
3
```

The above example is very statically defined. Suppose we need all numbers from one to hundred, then 97 instructors need to be added.

## Example:

Print all values from 1 to 10.

```
Module Example
    Sub Main()
        Dim value As Integer
        '
        For value = 1 To 10
            Console.WriteLine(value)
        Next
        '
        Console.ReadLine()
    End Sub
End Module
```

## Output :

```
1
2
3
4
5
6
7
8
9
```

An optional Step clause can be used to divert from the default stepvalue 1.

```
Module Example
  Sub Main()
    Dim value As Integer
    '
    For value = 2 To 10 Step 2
      Console.WriteLine(value)
    Next
    '
    Console.ReadLine()
  End Sub
End Module
```

Output :

```
2
4
6
8
10
```

Negative stepvalue are possible. But make sure the startvalue is more than the endvalue, otherwise the body would never execute.

```
Module Example
  Sub Main()
    Dim value As Integer
    '
    For value = 10 To 0 Step -2
      Console.WriteLine(value)
    Next
    '
    Console.ReadLine()
  End Sub
End Module
```

## LOOPING STRUCTURE

It executes a series of statements as long as a given condition is True.

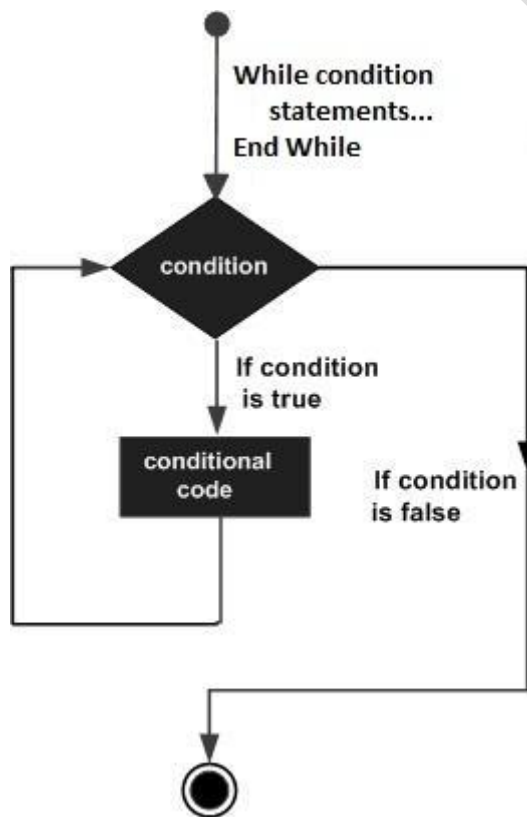
The syntax for this loop construct is:

```
While condition
  [ statements ]
  [ Continue While ]
  [ statements ]
  [ Exit While ]
  [ statements ]
End While
```

Here, statement(s) may be a single statement or a block of statements. The condition may be any expression, and true is logical true. The loop iterates while the condition is true.

When the condition becomes false, program control passes to the line immediately following the loop.

### Flow Diagram:



Here, key point of the *While* loop is that the loop might not ever run. When the condition is tested and the result is false, the loop body will be skipped and the first statement after the while loop will be executed.

## Example

```
Module loops
  Sub Main()
    Dim a As Integer = 10
    ' while loop execution '
    While a < 20
      Console.WriteLine("value of a: {0}", a)
      a = a + 1
    End While
    Console.ReadLine()
  End Sub
End Module
```

When the above code is compiled and executed, it produces the following result:

```
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```

```
Module Module1
  Sub Main()
    Dim a As Integer = 1
    Dim b As Integer
    ' while loop execution '
    While a < 21
      b = a * 9
      Console.WriteLine("9 * {0} = {1}", a, b)
      a = a + 1
    End While
    Console.ReadLine()
  End Sub
End Module
```

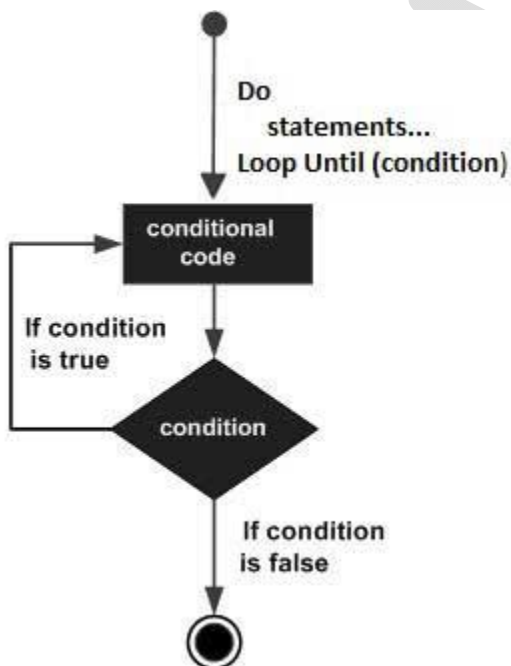
## Do Loop

It repeats the enclosed block of statements while a Boolean condition is True or until the condition becomes True. It could be terminated at any time with the Exit Do statement.

The syntax for this loop construct is:

```
Do { While | Until } condition
    [ statements ]
    [ Continue Do ]
    [ statements ]
    [ Exit Do ]
    [ statements ]
Loop
-or-
Do
    [ statements ]
    [ Continue Do ]
    [ statements ]
    [ Exit Do ]
    [ statements ]
Loop { While | Until } condition
```

### Flow Diagram:





## Example:

```
Module loops
  Sub Main()
    ' local variable definition
    Dim a As Integer = 10
    'do loop execution
    Do
      Console.WriteLine("value of a: {0}", a)
      a = a + 1
    Loop While (a < 20)
    Console.ReadLine()
  End Sub
End Module
```

When the above code is compiled and executed, it produces the following result:

```
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```

The program would behave in same way, if you use an Until statement, instead of While:

```
Module loops
  Sub Main()
    ' local variable definition
    Dim a As Integer = 10
    'do loop execution
    Do
      Console.WriteLine("value of a: {0}", a)
      a = a + 1
    Loop Until (a = 20)
    Console.ReadLine()
  End Sub
End Module
```

When the above code is compiled and executed, it produces the following result:

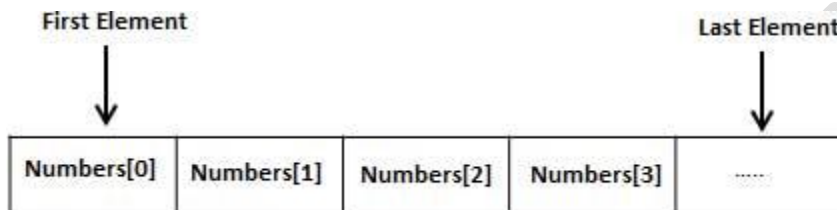
```
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
value of a: 16
value of a: 17
value of a: 18
```

## Array in VB.NET

### Arrays:

An array stores a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type.

All arrays consist of contiguous memory locations. The lowest address corresponds to the first element and the highest address to the last element.



### Creating Arrays in VB.Net

To declare an array in VB.Net, you use the Dim statement. For example,

```
Dim intData(30) ' an array of 31 elements
Dim strData(20) As String ' an array of 21 strings
Dim twoDarray(10, 20) As Integer 'a two dimensional array of integers
Dim ranges(10, 100) 'a two dimensional array
```

You can also initialize the array elements while declaring the array. For example,

```
Dim intData() As Integer = {12, 16, 20, 24, 28, 32}
Dim names() As String = {"Karthik", "Sandhya", _
"Shivangi", "Ashwitha", "Somnath"}
Dim miscData() As Object = {"Hello World", 12d, 16ui, "A"c}
```

The elements in an array can be stored and accessed by using the index of the array.

Example: Declare an array listing 5 animals in a zoo (aardvark, bear, cuckoo, deer, and elephant) in alphabetical order.

```
dim zooanimals() as string = {"aardvark","bear","cow","deer","elephant"}
```

Write code to output the first and last animal

**Answer :**

```
console.writeline(zooanimals(0))  
console.writeline(zooanimals(4))
```

What is the output of the following code:

```
dim primes() as integer = {2,3,5,7,11,13,17,19,23}  
dim count = 8  
While count >= 0  
    console.write(primes(count) & ", ")  
    count = count - 1  
end while  
23,19,17,13,11,7,5,3,2
```

## Programming of Pre release material (SPECIMEN)

Module Module1

```
Sub Main()  
    Dim Name(30) As String  
    Dim Test1(30), Test2(30), Test3(30) As Single  
    Dim Student_Total(30) As Single  
    Dim Class_Total As Single  
    Dim Class_Average As Single  
    Dim highest_marks As Single  
    Dim highest_name As String  
  
    Class_Total = 0  
    highest_marks = 0  
  
    For Count = 1 To 3  
  
        Console.WriteLine("Student No. " & Count)  
        Console.Write("Enter Name Of Student: ")  
        Name(Count) = Console.ReadLine()  
        Console.Write("Enter Marks Of Test 1: ")  
        Test1(Count) = Console.ReadLine()  
  
        Do Until Test1(Count) <= 20  
            Console.WriteLine("Invalid Marks: Enter Marks out of 20")  
            Console.Write("Enter Marks Of Test 1: ")  
            Test1(Count) = Console.ReadLine()  
        Loop  
  
        Console.Write("Enter Marks Of Test 2: ")  
        Test2(Count) = Console.ReadLine()  
  
        Do Until Test2(Count) <= 25  
            Console.WriteLine("Invalid Marks: Enter Marks out of 25")  
            Console.Write("Enter Marks Of Test 2: ")  
            Test2(Count) = Console.ReadLine()  
        Loop  
  
        Console.Write("Enter Marks Of Test 3: ")  
        Test3(Count) = Console.ReadLine()  
  
        Do Until Test3(Count) <= 35  
            Console.WriteLine("Invalid Marks: Enter Marks out of 35")  
            Console.Write("Enter Marks Of Test 3: ")  
            Test3(Count) = Console.ReadLine()  
        Loop  
  
        Student_Total(Count) = Test1(Count) + Test2(Count) + Test3(Count)  
  
        If Student_Total(Count) > highest_marks Then  
            highest_marks = Student_Total(Count)  
            highest_name = Name(Count)  
        End If  
    End For  
End Sub
```

```
Class_Total = Class_Total + Student_Total(Count)

    Console.WriteLine(Student_Total(Count))
Next

    Console.WriteLine("Name " & " | " & "Total Marks")
    For Count = 1 To 3
        Console.WriteLine(Name(Count) & " | " &
Student_Total(Count))
    Next
    Console.WriteLine("Name of Student with highest marks: " &
highest_name)
    Console.WriteLine("Highest marks: " &
highest_marks)
    Class_Average = Class_Total / 3

    Console.WriteLine("The average of class is: " & Class_Average)

    Console.ReadKey()

End Sub
End Module
```

## Algorithm of Pre Release 2015

Set Name [1:30]

Set FirstWeight [1:30]

Set LastWeight [1:30]

Set Difference [1:30]

FOR Count ← 1 to 30

INPUT Name

Name [Count] ← Name

INPUT FirstWeight

IF FirstWeight < 25 OR FirstWeight > 150 THEN

PRINT "Invalid weight, Try again"

Count ← Count - 1

ELSE

FirstWeight [Count] ← FirstWeight

ENDIF

PRINT Name [Count], FirstWeight [Count]

NEXT Count

FOR Count ← 1 to 30

INPUT LastWeight

IF LastWeight < 25 OR LastWeight > 150 THEN

PRINT "Invalid weight, Try again"

Count ← Count - 1

ELSE

LastWeight [Count] ← LastWeight

ENDIF

NEXT Count

FOR Count ← 1 to 30

Diff [Count] ← LastWeight [Count] - FirstWeight [Count]

IF Diff [Count] > 2.5 THEN

PRINT "Rise in weight of Student" Name [Count] "is" Diff [Count]

ENDIF

IF Diff [Count] < - 2.5 THEN

PRINT "Fall in weight of Student" Name [Count] "is" Diff [Count]

ENDIF

NEXT Count

## Programming of Pre Release 2015 On VB.NET

### Task 1

A school keeps records of the weights of each pupil. The weight, in kilograms, of each pupil is recorded on the first day of term. Input and store the weights and names recorded for a class of 30 pupils. You must store the weights in a one-dimensional array and the names in another one-dimensional array. All the weights must be validated on entry and any invalid weights are rejected. You must decide your own validation rules. You may assume that the pupils' names are unique. Output the names and weights of the pupils in the class.

### Task 2

The weight, in kilograms, of each pupil is recorded again on the last day of term. Calculate and store the difference in weight for each pupil.

### Task 3

For those pupils who have a difference in weight of more than 2.5 kilograms, output with a suitable message, the pupil's name, the difference in weight and whether this is a rise or a fall.

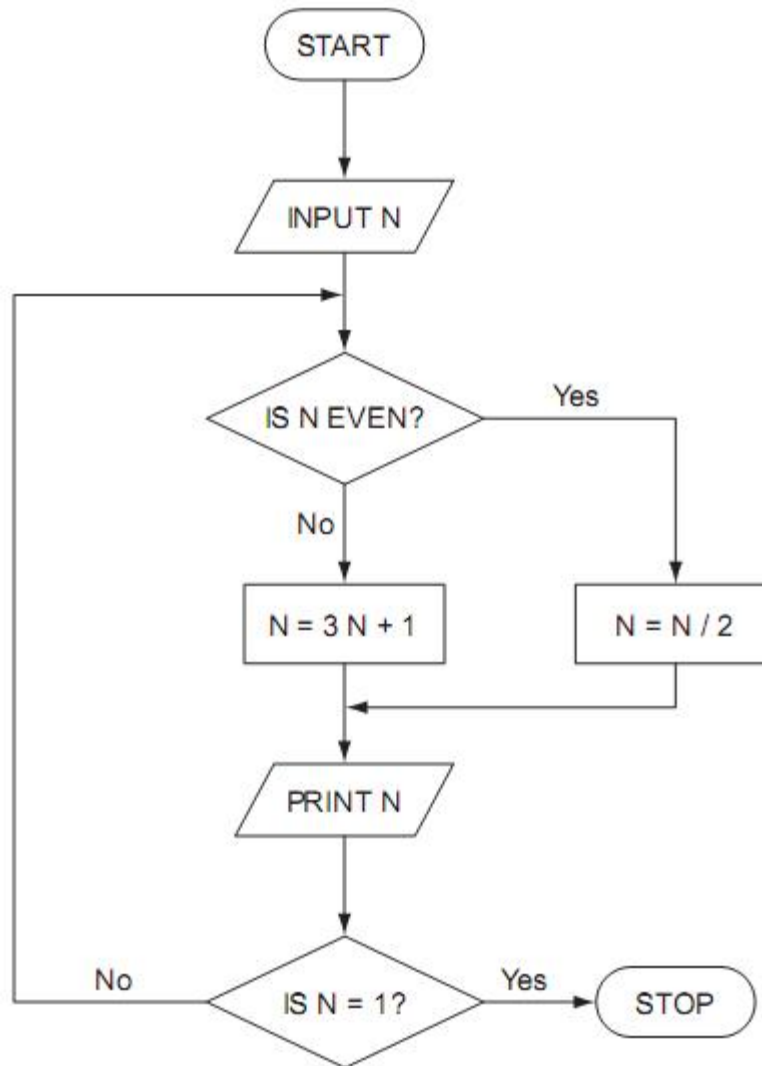
### Solution:

# **FLOW CHARTS WORKSHEETS**



May/June 2006

Q1)



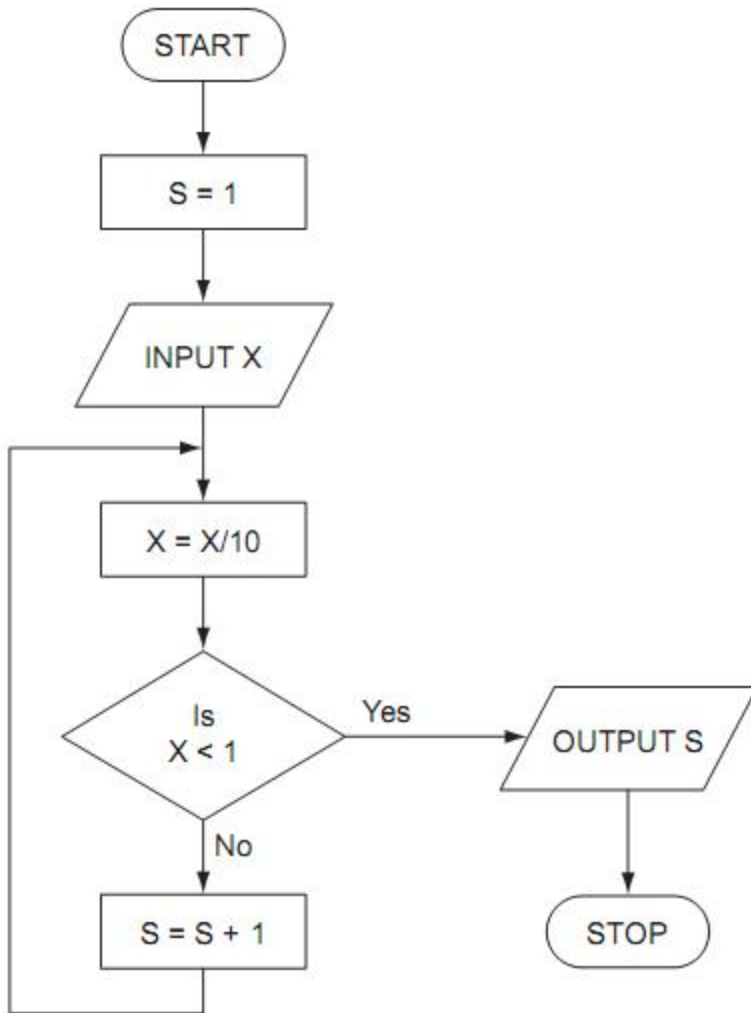
Trace the flow chart using the numbers 2 and 3. Write down each of the values of N in the order that they are printed out.

(a) 2 .....

(b) 3 .....

May/June 2007

Q2) Study the following flowchart very carefully.

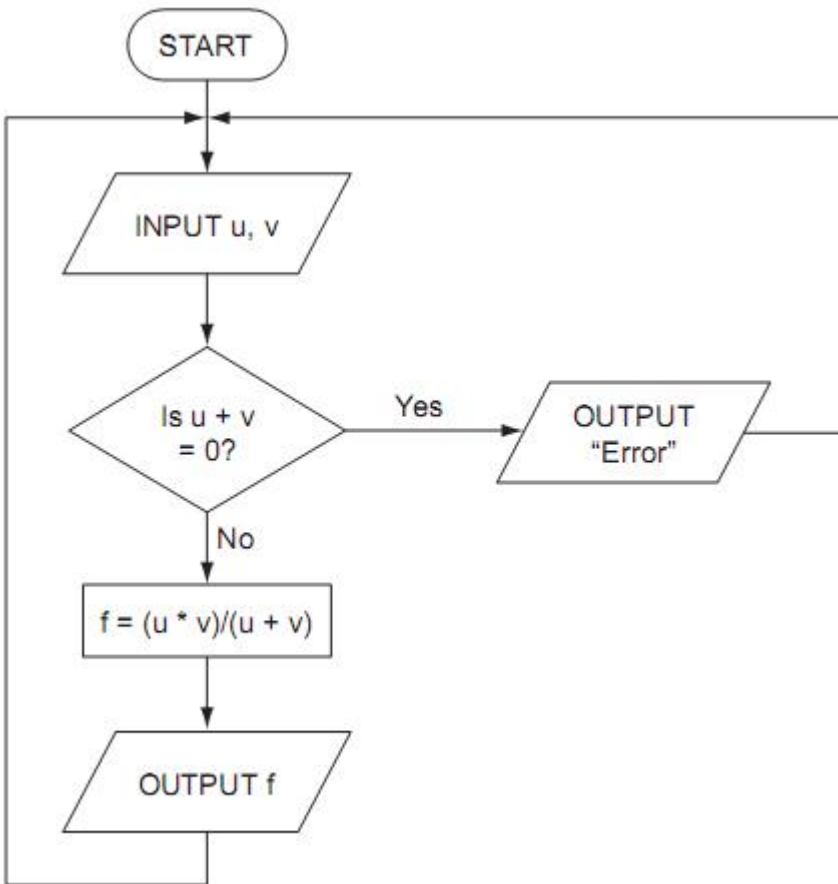


Complete the following table showing the expected output from the flowchart for the three sets of input data:

INPUT X	OUTPUT S
48	
9170	
- 800	

May/June 2008

Q3) The following flowchart inputs two numbers, carries out a calculation and then outputs the result.



(a) Complete the following table for the three sets of input data.

INPUT		OUTPUT
u	v	
5	5	
6	-6	
12	4	

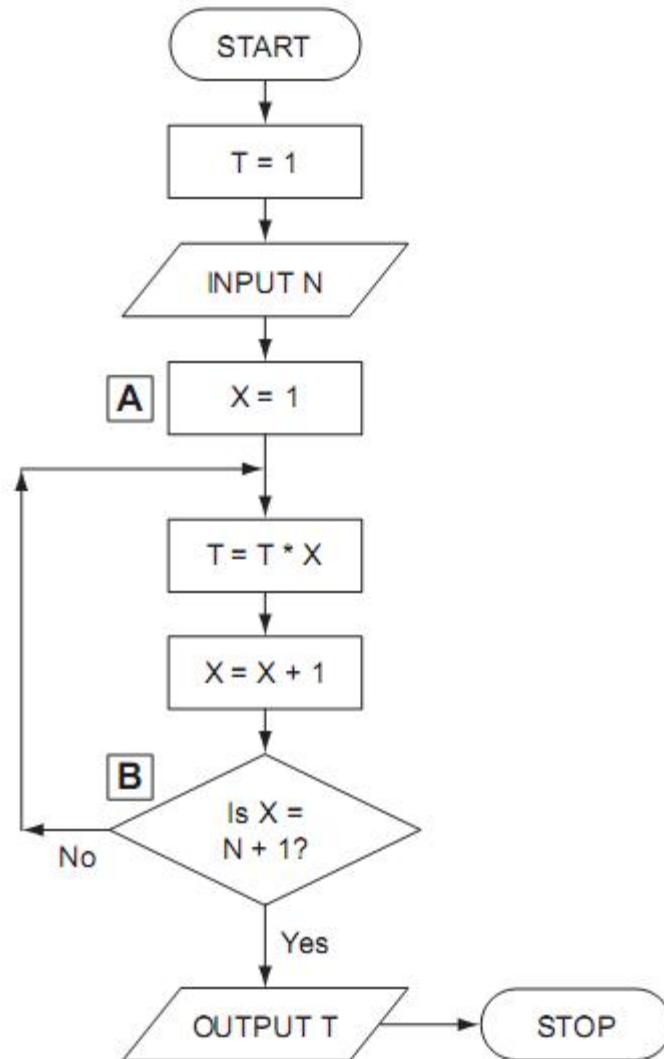
(b) The above algorithm has been placed in a library of routines. Give one advantage of doing this.

.....

.....

May/June 2009

Q4)



(a) Complete the table to show what outputs you would expect for the two inputs.

Input N	Output T
5	
1	

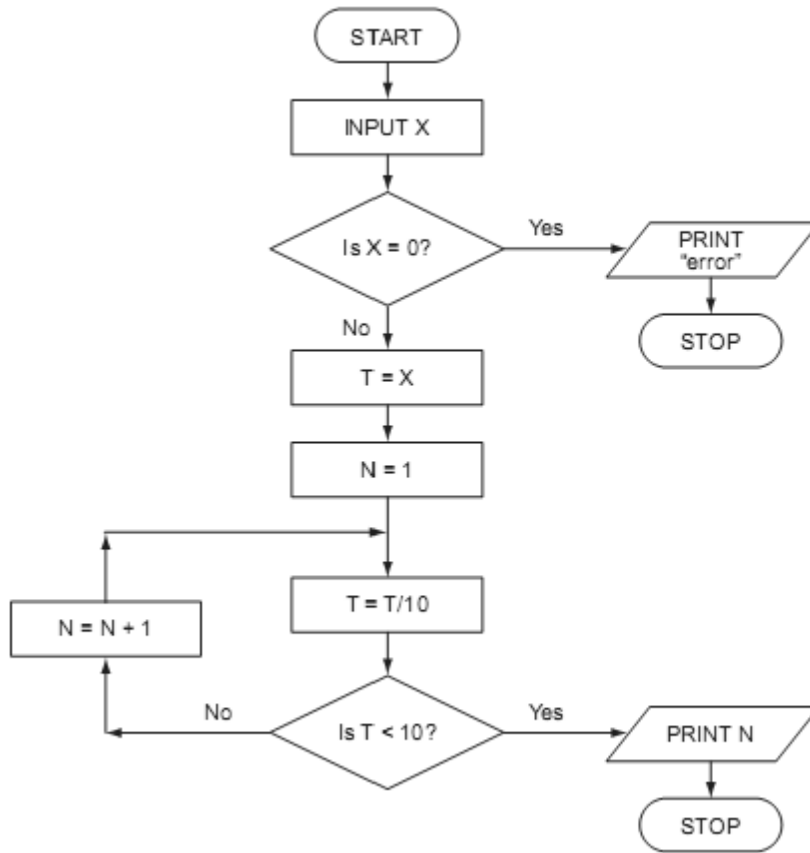
(b) Write down a possible LOOP construct for the section A to B in the flowchart using pseudocode.

.....

.....

May/June 2010

Q5) Study the following flowchart very carefully:

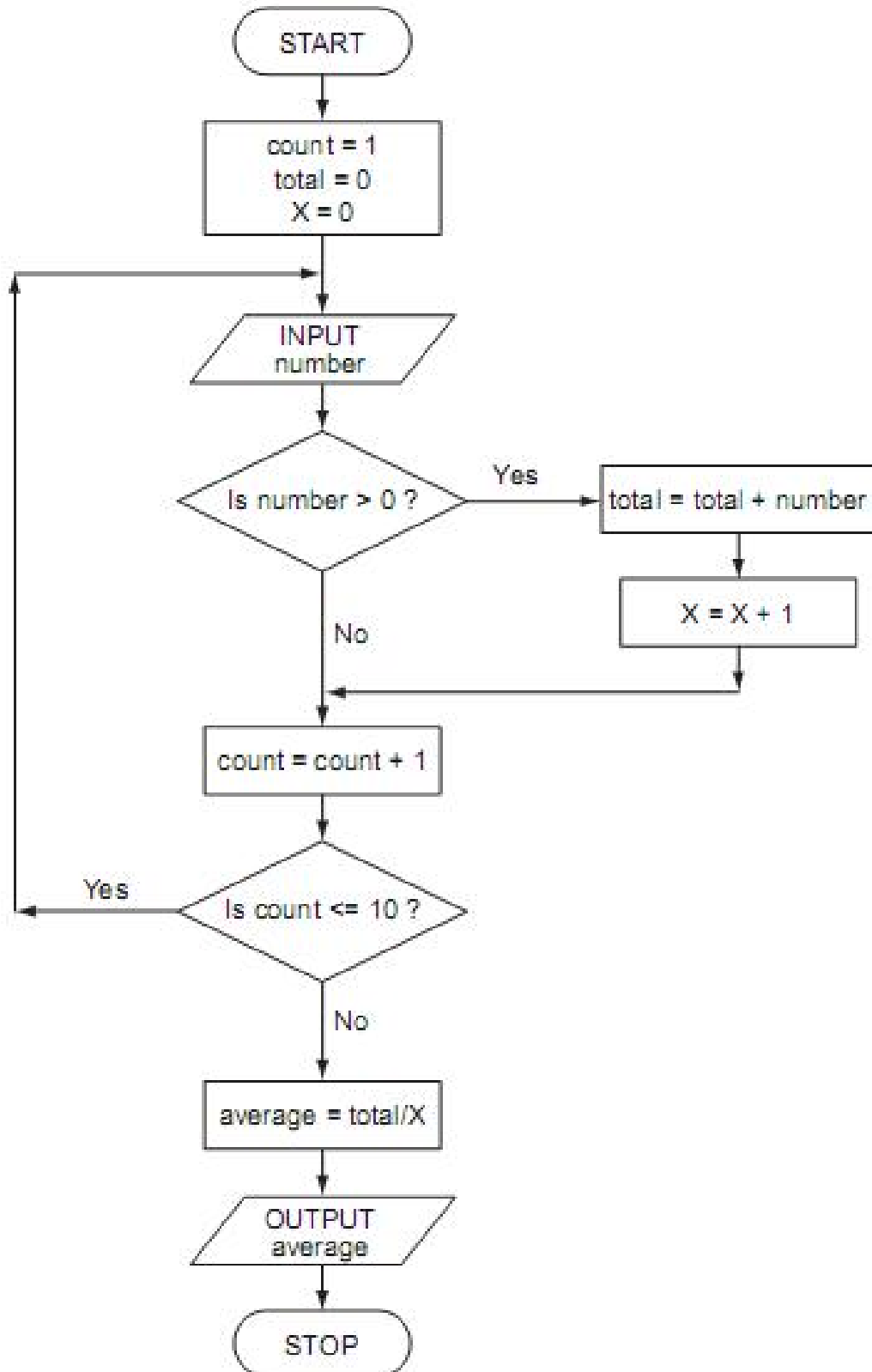


What output would you expect if the following data was input into the flowchart?

X	OUTPUT
-150	
540	
0	

May/June 2011

Q6) Study the following flowchart very carefully:



(a) Complete the trace table for the following data set:

15, -2, 0, 8, 0, 21, -8, -12, 1, 25

count	number	total	X	average	OUTPUT

[4]

(b) What is the purpose of this flowchart?

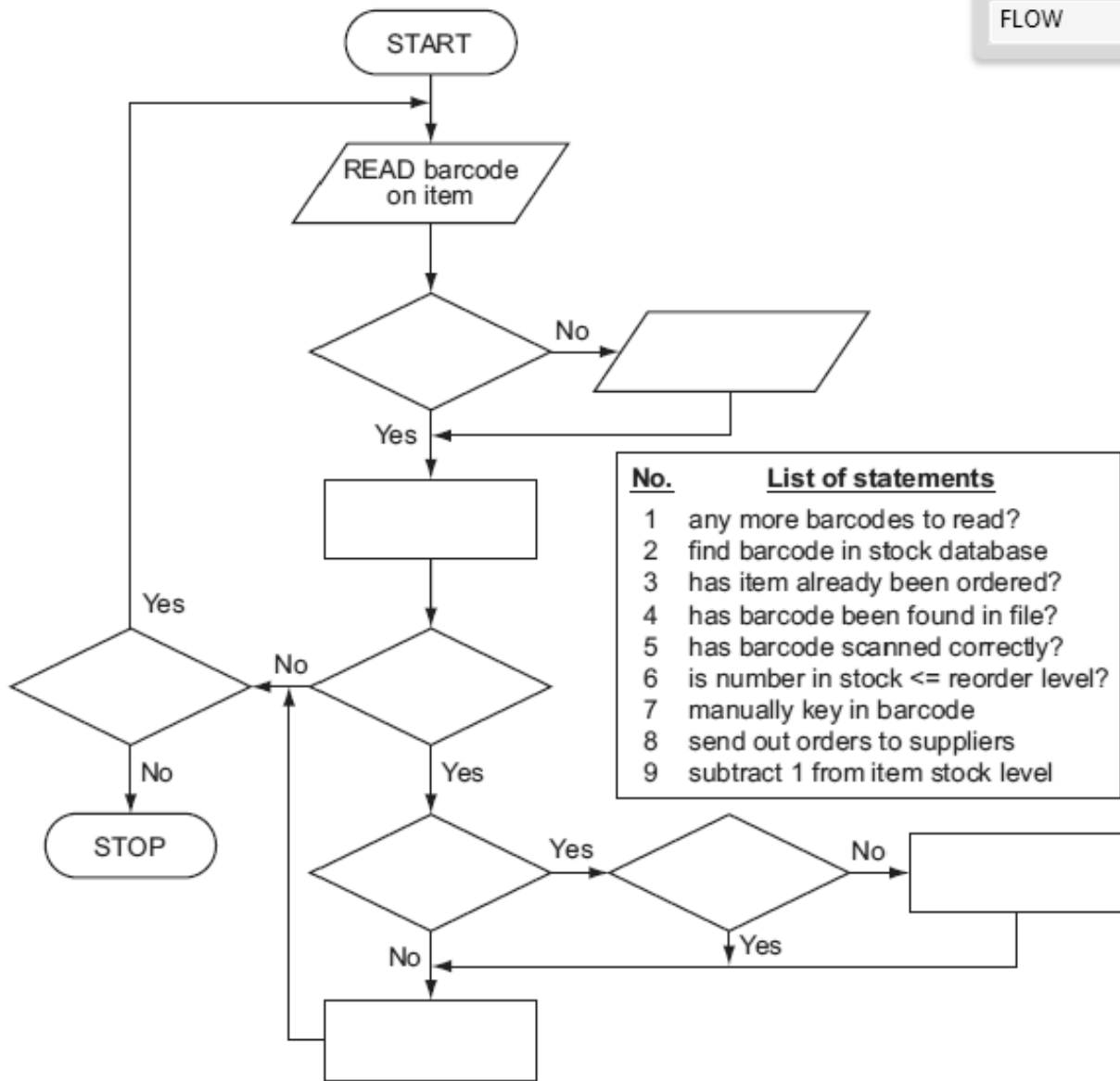
.....

.....

October/November 2011

Q7) The following flowchart shows how barcodes are used at the point of sale in an automatic stock control system.

Select statements from the list below, using numbers only, to complete the flowchart.

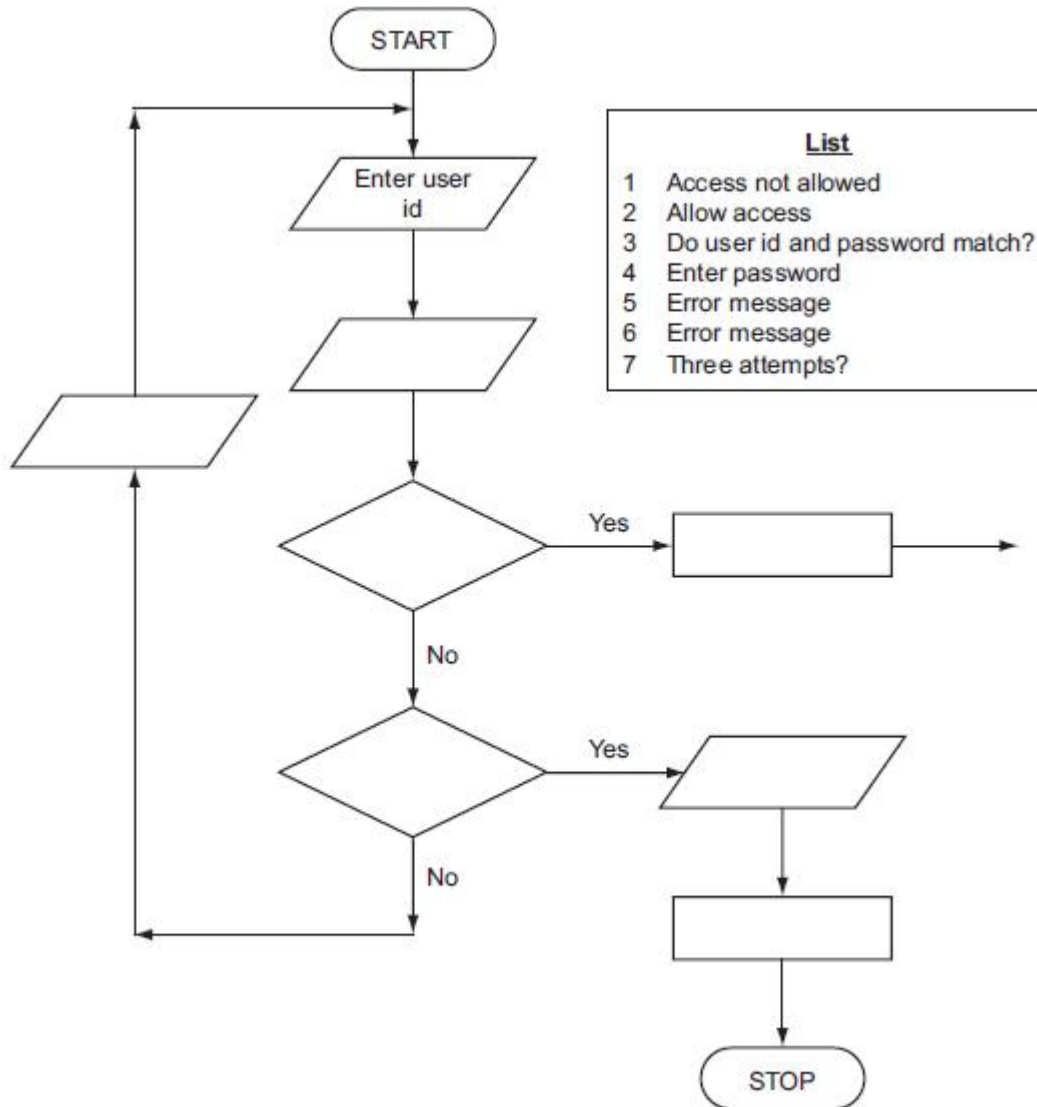


[5]



October/November 2010

(a) To log on to a computer, a user needs to type in a user id followed by a password; these should match up. Only three attempts are allowed. The flowchart below shows the log on procedure. Several boxes have been left blank. Complete the flowchart using items from the list.



[3]

# **PSEUDOCODE WORKSHEETS**

**MAY/JUNE SESSION 2003**

Q1) A school wants to monitor the number of hours spent by a class of 30 students on the Internet.

Using pseudocode or otherwise, write an algorithm which will;

- for each student, record the times logged on and logged off
- calculate the length of time each student spends online
- calculate and output the average length of time per day spent by each student on the Internet.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

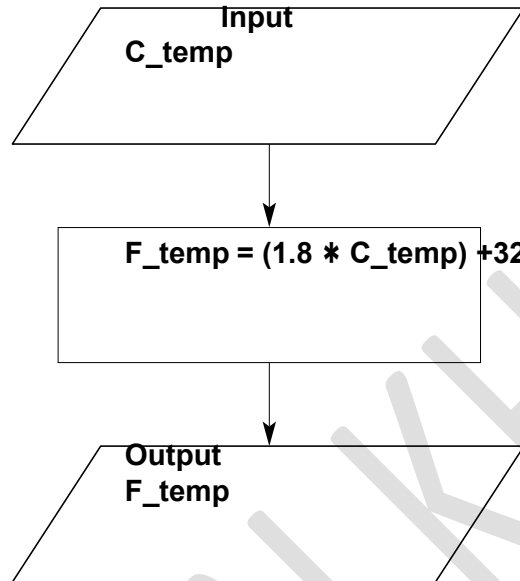
.....

.....

.....

**MAY/JUNE SESSION 2004**

Q2) Read this algorithm. The algorithm converts a temperature from degrees Centigrade to degrees Fahrenheit.



(a) Write down the output for each of the following inputs:

(i) 1

.....

(ii) 5

.....

[1]

- (b)** Using pseudocode, or otherwise, write an algorithm that will input the hourly temperatures for one day in Centigrade and print out in Fahrenheit
- the maximum temperature
  - the minimum temperature
  - the average temperature for

that day.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

**MAY/JUNE SESSION 2005**

Q3) The following algorithm contains an error.

1. SET  $X = 1$
2. REPEAT
3.      $X = X + 2$
4.     Print  $X$
5. UNTIL  $X = 10$

(a) Trace the algorithm and explain what the error is.

.....  
.....  
.....  
.....  
.....

[2]

(b) Write an algorithm which uses a While..Do..Endwhile loop and outputs the numbers 2, 4, 6 and 8.

.....  
.....  
.....  
.....  
.....  
.....  
.....  
.....  
.....

[3]

**Q4)** Using pseudocode or otherwise, write an algorithm that will input 25 marks and output the number of DISTINCTION, MERIT, PASS or FAIL grades.

A mark greater than 69 will get a DISTINCTION, a mark between 69 and 60 (inclusive) will get a MERIT and a mark between 59 and 50 (inclusive) will get a PASS.

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

IMRAN KHAN

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

[6]

**May/June 2006**

Q5) A formula for calculating the body mass index (BMI) is:

$$\text{BMI} = \frac{\text{weight in kilograms}}{(\text{height in metres}) \times (\text{height in metres})}$$

Calculate the BMI for a person whose weight is 80kg and height is 2 metres.

.....

[1]

(b) Using pseudocode or otherwise, write an algorithm that will input the ID, weight (kg) and height (m) of 30 students, calculate their body mass index (BMI) and output their ID, BMI and a comment as follows:

A BMI greater than 25 will get the comment 'OVER WEIGHT', a BMI between 25 and 19 (inclusive) will get 'NORMAL' and a BMI less than 19 will get 'UNDER WEIGHT'.





**May/June 2007**

Q6) A company has 5000 CDs, DVDs, videos and books in stock. Each item has a unique 5-digit code with the first digit identifying the type of item, i.e.

- 1 = CD
- 2 = DVD
- 3 = video
- 4 = book

For example, for the code 15642 the 1 identifies that it is a CD, and for the code 30055 the 3 identifies that it is a video.

Write an algorithm, using pseudocode or otherwise, that

- Inputs the codes for all 5000 items
- Validates the input code
- Calculates how many CDs, DVDs, videos and books are in stock
- Outputs the four totals.

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

[6]

Q7) Algorithms and programs use loops to control the number of times a particular procedure is used.

Two methods are repeat ... until and for ... to.

(a) Write a procedure using both these loop methods to input 20 numbers into a variable called x.

(i) repeat ... until

.....

..

.....

..

.....

..

.....

(ii) for ... to

.....

..

.....

..

.....  
..

.....  
(b) Name another loop structure.

.....  
..

.....

IMRAN KHAN

Q8) Customers can withdraw cash from an Automatic Teller Machine (ATM).

- withdrawal is refused if amount entered > current balance
- withdrawal is refused if amount entered > daily limit
- if current balance < \$100, then a charge of 2% is made
- if current balance  $\geq$  \$100, no charge is made

Write an algorithm which inputs a request for a sum of money, decides if a withdrawal can be made and calculates any charges. Appropriate output messages should be included.

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

[6]

IMRAN KHAN

Q9) A small airport handles 400 flights per day from three airlines:

FASTAIR	(code
FA)	
SWIFTJET	(code
SJ)	
KNIGHTAIR	(code
KA)	

Each flight is identified by the airline code and 3 digits. For example FA 156.

Write an algorithm, using pseudocode or otherwise, which monitors the 400 flights into and out of the airport each day. The following inputs, processing and outputs are all part of the monitoring process:

- input flight identification
- calculate number of flights per day for each of the three airlines
- output the percentage of the total flights per day by each airline
- any validation checks must be included

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..





**May/June 2010**

Q10) A group of students were monitoring the temperature every day over a one-year period.

Readings were taken ten times every day (you may assume a year contains 365 days).

Write an algorithm, using pseudocode or flowchart, which

- inputs all the temperatures (ten per day)
- outputs the highest temperature taken over the year
- outputs the lowest temperature taken over the year
- outputs the average temperature per day
- outputs the average temperature for the whole year

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

[6]

Q11) (a) Write an algorithm, using pseudocode or a flowchart, which:

- inputs 50 numbers
- outputs how many of the numbers were  $> 100$

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..

[3]

- (b) Write an algorithm, using pseudocode or a flowchart, which:
- inputs 100 numbers
  - finds the average of the input numbers
  - outputs the average

.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..

.....  
[3]

May/June 2011

Q12) Daniel lives in Italy and travels to Mexico, India and New Zealand. The times differences are:

<u>Country</u>	<u>Hours</u>	<u>Minutes</u>
Mexico	-7	0
India	+4	+30
New Zealand	+11	0

Thus, if it is 10:15 in Italy it will be 14:45 in India.

(a) Write an algorithm, using pseudocode or otherwise, which:

- Inputs the name of the country
- Inputs the time in Italy in hours (H) and minutes (M)
- Calculates the time in the country input using the data from the table
- Outputs the country and the time in hours and minutes

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....

..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
[4]

(b) Describe, with examples, two sets of test data you would use to test your algorithm.

1.....  
....

.....  
..

2.....  
....

.....  
.. [2]

**May/June 2012**

Q13) Write an algorithm, using pseudocode or a program flowchart only, which:

- inputs the population and land area for 500 countries,
- calculates the population density (i.e. population/land area) for every country,
- outputs the largest and smallest population density,
- outputs the average population for all 500 countries.

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

..

..

. [6]

**OCTOBER/NOVEMBER SESSION 2001**

Q14) This algorithm grades candidates on marks out of ten.

```
1  input a Mark
2      case Mark of
3          0, 1, 2, 3 : Grade = Fail
4          4, 5 : Grade = Pass
5          6, 7 : Grade = Merit
6          8, 9, 10 : Grade = Distinction
7      otherwise Mark = -1
8  endcase
9  if Mark = -1 then
10     print 'Not a valid mark'
11 else output Grade, 'Grade'
```

(a) Dry run the algorithm for each of the following data and complete the table.

INPUT	OUTPUT
<b>0</b>	
<b>5</b>	
<b>99</b>	

[3]

(b) Write down two instructions which could be inserted between lines 1 and 2

to allow the algorithm to deal with marks out of 100.

.....  
...  
.....  
...  
.....  
...  
.....

[2]

Q15) Employees of a shop are entitled to a discount of 10% on the value of goods bought from the shop. However if an employee has worked at the shop for five or more years they are entitled to a discount of 20%. Only employees are allowed discounts. The discount on electrical goods is fixed at only 10%.

Using pseudocode or otherwise, write an algorithm which will determine what discount applies when any person buys an item.

.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....



..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

Q16) part of a question

(c) Using an example, or otherwise, show the difference between a

**Repeat ...Until**  
construct and an

**If ... then ... Else ... Endif** construct.

**Repeat ...Until**

.....

..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....

**If ... then ... Else ... Endif**

.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....

**OCTOBER/NOVEMBER SESSION 2002**

Q17) Read this algorithm.

set **Total\_1** to zero

set **Total\_2** to zero

set **Counter** to one

while **Counter** < eight

**Counter = Counter + 1**

input **Number**

if **Number** > zero then **Total\_1 = Total\_1 + Number**

if **Number** < zero then **Total\_2 = Total\_2 + Number**

endwhile

output **Total\_1** output **Total\_2**

(a) Write down the output if the following set of numbers are input.

4, 1, -3, 2, -5, 0, 6

.....  
[2]

(b) Modify the algorithm so that it will accept any number of numbers, the input is terminated by a rogue value and the output is the **Total** of all the numbers input except the rogue value.

.....  
.....  
.....  
.....  
.....  
.....  
.....  
.....  
.....  
.....  
.....  
.....  
.....  
.....  
.....  
.....  
.....  
.....  
.....  
.....

...

.....

...

.....

...

.....

...

.....

...

.....

...

.....

[4]

Q18)

Using pseudo code or otherwise, write an algorithm which will input any three different numbers and then print them out in ascending order.

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

[6]

### October/November 2003

Q19) The following algorithm inputs air speeds (which must be in multiples of 100) and outputs a suitable message.

```
1  input a speed
2  whole = speed/100
3      case whole of
4          0,1,2 : result = slow
5          3, 4, 5, 6 : result = normal
6          7, 8, 9 : result = high
7          otherwise whole = -1
8  endcase
9  if whole = -1 then
10     output "abnormal reading"
11 else output result, "speed"
```

(a) Dry run the above algorithm for the following Input data and complete the Output column in the table:

Input	Output
150	
400	
800	

[3]

(b) State what would happen if line 2 had been missed out of the algorithm.

.....

..

.....

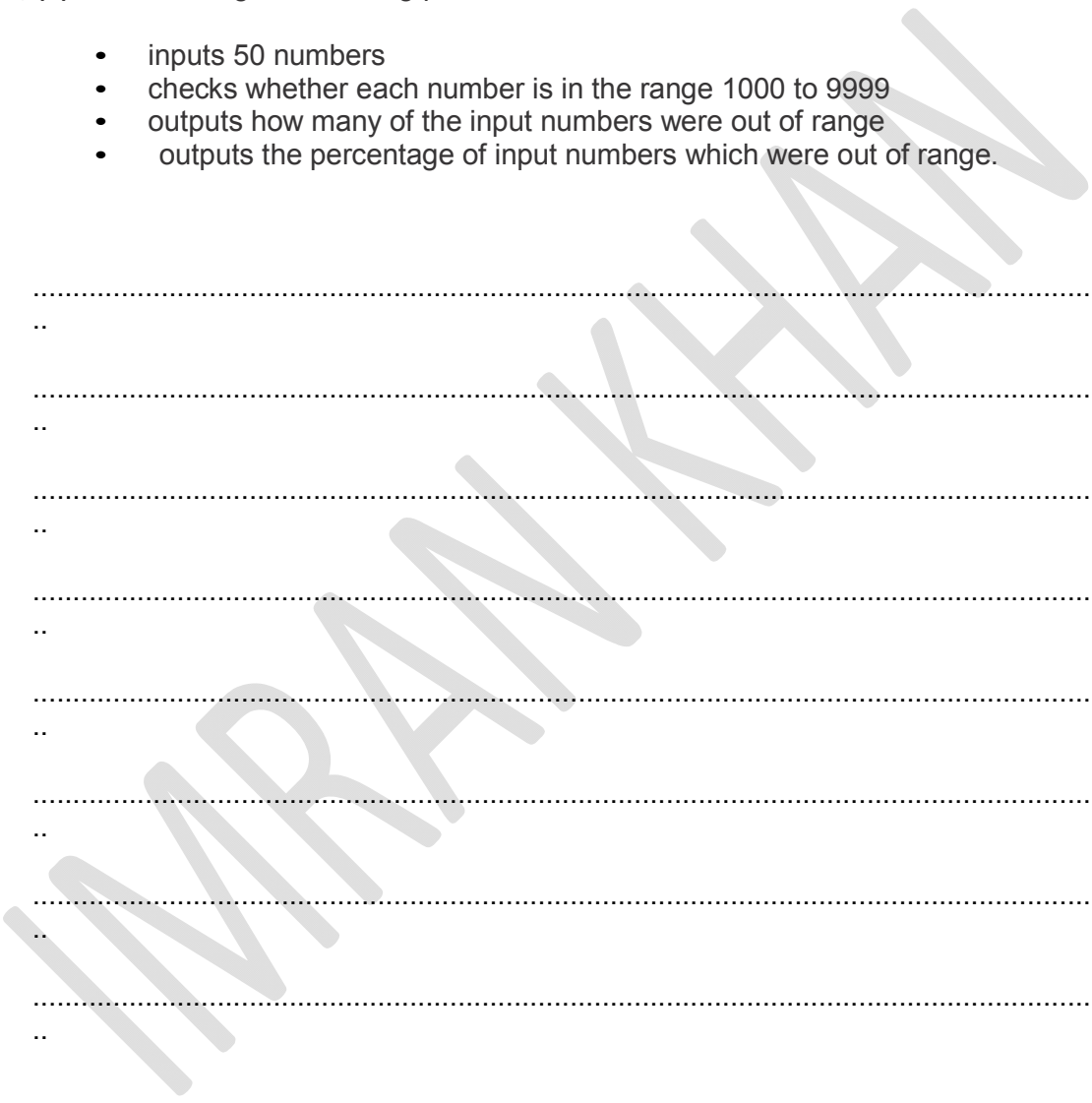
..

.....  
..

.....  
[2]

Q20) (a) Write an algorithm, using pseudocode or otherwise, which;

- inputs 50 numbers
- checks whether each number is in the range 1000 to 9999
- outputs how many of the input numbers were out of range
- outputs the percentage of input numbers which were out of range.



.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..

.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..

[6]

(b) Describe, using examples, **two** validation checks other than range check which could be carried out on the input numbers.

Validations check 1.....

Example.....

.....  
.....

Validations check 2.....

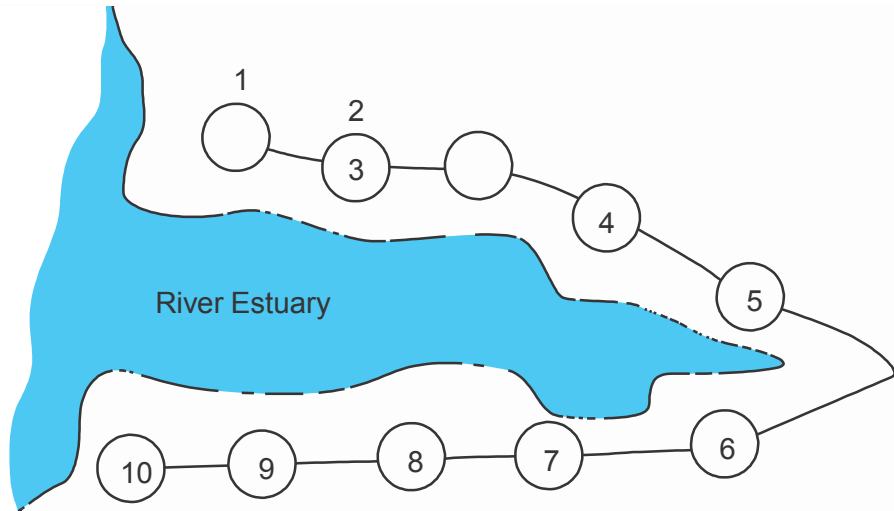
Example.....

.....  
.....

[4]

### October/November 2004

Q21) The following diagram shows a rail network.



The rail network consists of 10 stations. The fare between each station is \$2. There is a 10% discount when 3 or more passengers travel together. Tickets can be purchased at any station using automated terminals.

Using pseudocode, or otherwise, write an algorithm for the automated terminals to:

- input the starting station number, the destination station number and the number of passengers
- calculate the total fare and output the amount to be paid
- calculate the change (if any)
  - issue the rail ticket(s) and change

.....  
..

.....  
..

.....  
..

.....  
..



..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....

IMRAN KHAN

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

[6]

IMRAN KHAN

### October/November 2005

Q22) A school uses a computer to store student marks obtained in an end of term mathematics exam. There are 150 students doing the exam and the maximum mark is 100.

Write an algorithm, using pseudocode or otherwise, which

- inputs the marks for all students
- checks if each mark is in the correct range and, if not, the mark is re-input
- outputs the smallest mark
- outputs the highest mark
- outputs the average mark for the exam.

IMRAN KHAN

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

[6]

**October/November 2006**

Q23) A computer program is required which inputs 10 numbers, multiplies them together and finally outputs the answer (the product). The following algorithm has been written to do this.

```
1 count = 0
2 product = 0
3 while count <= 10 do
4     input number
5     product = product * number
6     count = count + 1
7     print product
8 endwhile
```

(a) There are three errors in the algorithm. Locate and describe these errors.

1

.....

.....

.....

2

.....

.....  
.....  
3 .....

..... [3]

(b) A while ... do loop has been used in the algorithm. State another type of loop that could have been used.

.....  
.....  
..... [1]

Q24)

Temperatures ( C) are being collected in an experiment every hour over a 200 hour period.

Write an algorithm, using pseudocode or otherwise, which inputs each temperature and outputs.

- how many of the temperatures were above 20°C
- how many of the temperatures were below 10°C
- the lowest temperature that was input

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

[6]

**October/November 2007**

Q25) (a) Fuel economy for a car is found using the formula:

$$\text{Fuel Economy} = \frac{\text{Distance Travelled (km)}}{\text{Fuel Used (litres)}}$$

What would be the Fuel Economy of a car travelling 40 km on 10 litres of fuel?

..... [1]

(b) The Fuel Economy for 1000 cars is to be calculated using the formula in Question 25(a).

Write an algorithm, using pseudocode or otherwise, which inputs the Distance Travelled (km) and the Fuel Used (litres) for 1000 cars. The Fuel Economy for each car is then calculated and the following outputs produced:

- Fuel Economy for each car
- average (mean) Fuel Economy for all of the cars input

- the best Fuel Economy (i.e. highest value)
- the worst Fuel Economy (i.e. lowest value)

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

[6]



**October/November 2008**

Q26) The manufacturing cost of producing an item depends on its complexity. A company manufactures three different types of item, with costs based on the following calculations:

- Item type 1: item cost = parts cost \* 1.5
- Item type 2: item cost = parts cost \* 2.5
- Item type 3: item cost = parts cost \* 5.0

The company makes 1000 items per day.

Write an algorithm, using pseudocode, flowchart or otherwise, which

- inputs the item type and parts cost of each item
- outputs the item cost for each item
- calculates and outputs the average (mean) item cost per day (based on 1000 items being made).

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
[6]

**October/November 2009**

Q27) A car's speed is measured between points A and B, which are 200 km apart.



The final speed of the car is calculated using the formula:

$$\text{Final Speed} = \frac{200}{\text{Time (hours)}}$$

What is the final speed of a car if it takes 2 hours to get from A to B?

.....  
..

.....  
..

(b) Write an algorithm, using pseudocode or otherwise,  
which inputs the times for 500 cars, calculates the final speed of each car using the formula in part (a),  
and then outputs:

- the final speed for ALL 500 cars
- the slowest (lowest) final speed
- the fastest (highest) final speed
- the average final speed for all the cars.

.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..

.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
..  
.....  
[6]

**October/November 2010**

Q28)

The following algorithm inputs 20 numbers and outputs how many numbers were positive(> 0) and how many numbers were negative (< 0).

```
1  negative = 1
2  positive = 1
3  for count = 1 to 20 do
4      input number
5      if number < 0 then negative = negative + 1
6      if number > 0 then positive = positive + 1
7      count = count + 1
8      print negative, positive
9  next count
```

There are three different errors in this algorithm.

Locate each error and give the reason why you think it is an

error. Error 1 .....

Reason 1 .....

Error 2 .....

Reason 2 .....

Error 3 .....

Reason 3 .....

[6]

Q29) A school is doing a check on the heights and weights of all its students. The school has 1000 students.

Write an algorithm, using pseudocode or a flowchart, which

- inputs the height and weight of all 1000 students
- outputs the average (mean) height and weight
- includes any necessary error traps for the input of height and weight

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

[6]

Q30) (a) Write an algorithm, using pseudocode or flowchart only, which:

- inputs three numbers
- outputs the largest of the three numbers

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

..

.....

..

.....

..

.....

[3]

(b) Write an algorithm, using pseudocode or flowchart only, which:

- inputs 1000 numbers
  - outputs how many of these numbers were whole numbers (integers)
- (You may use  $\text{INT}(X)$  in your answer e.g.  $Y = \text{INT}(3.8)$  gives the value  $Y = 3$ )

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

[4]

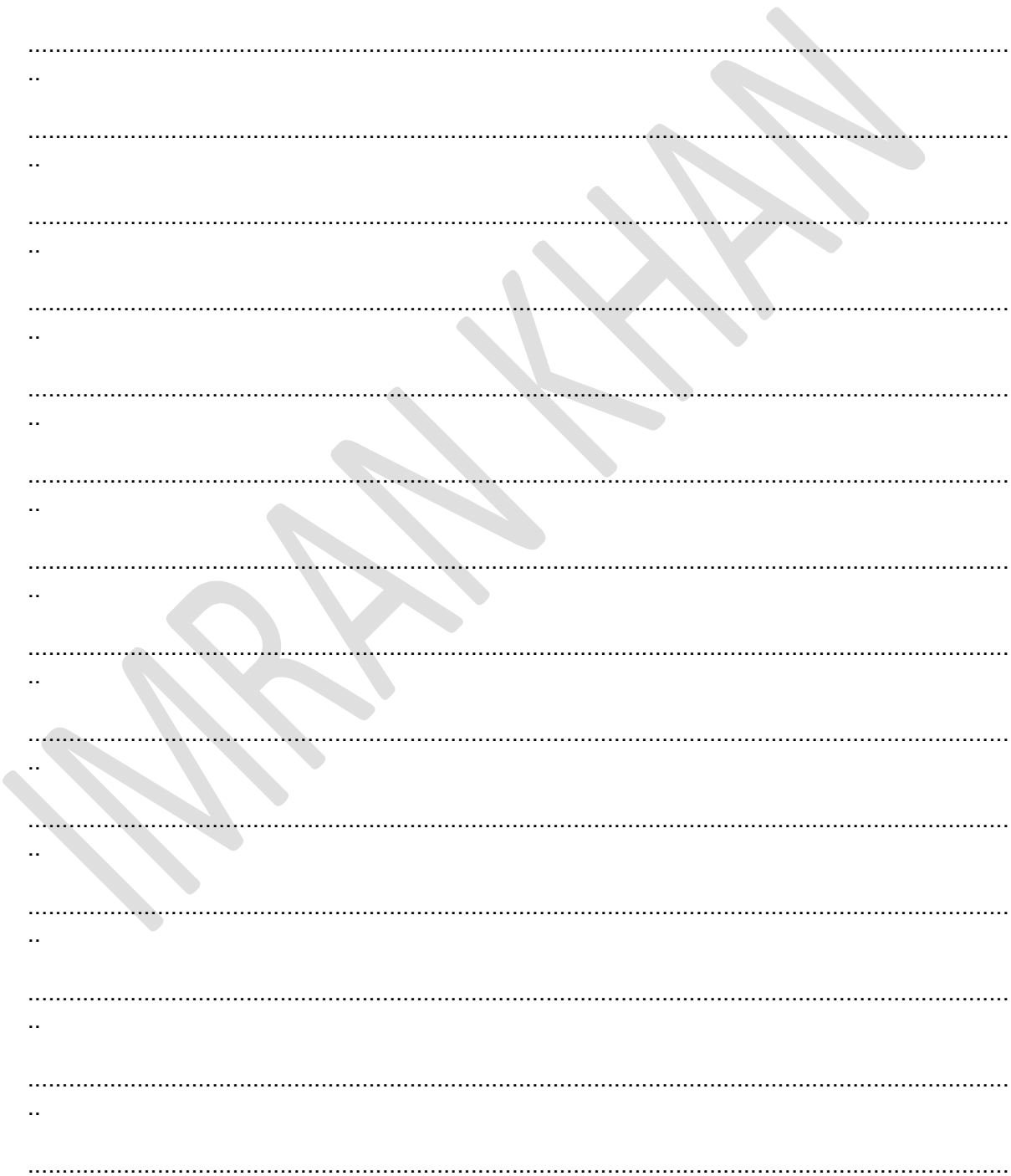
IMRAN KHAN



Q31) The weather conditions in a town are being monitored over a year (365 days). The values recorded per day are weather type and temperature (e.g. CLOUDY, 25).

Write an algorithm, using pseudocode or flowchart only, which:

- inputs the weather type and temperature for each day
- outputs the number of days that were CLOUDY, RAINING, SUNNY or FOGGY
- outputs the highest recorded temperature for the year
- outputs the lowest recorded temperature for the year



A series of horizontal dotted lines are provided for writing an algorithm or pseudocode, starting from the top of the page and extending down to the bottom.

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

[6]

IMRAN KHAN

Q32) Write an algorithm, using pseudocode or a program flowchart only, that:

- inputs a series of positive numbers (-1 is used to terminate the input),
- outputs how many numbers were less than 1000 and
- outputs how many numbers were greater than 1000.

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

.....

..

[4]

(b) Write an algorithm, using pseudocode or a program flowchart only, that

- inputs fifty numbers each as 4 separate digits, for example: 1 5 4 1
- outputs the percentage of numbers that were palindromes.

(note: a palindrome reads the same way backwards or forwards. For example, 1331 is a palindrome but 1541 is not).

Use separate variables to store the separate digits of a number (for example D1, D2, D3, D4).

.....

..

.....

..

.....

..

.....

..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

.....  
..

[4]

IMRAN KHAN

# **FLOW CHARTS WORKSHEETS**

Q1)

(a) 1

(b)  $\overleftarrow{10, 5,}$   $\overleftarrow{16, 8, 4, 2, 1}$   
one mark                      one mark

Q2)

(a) 2

4

1

Q3)

(a) 2.5

Error

3 [3]

(b) Any one from:

would be fully tested

doesn't need to be re-written each time section of program needed [1]

Q3)

(a) 120

1 [2]

(b) for X = 1 to N + 1    OR repeat OR while X <> N + 1 do

(T = T \* X)    (T = T \* X)    (T = T \* X)

X = X + 1    X = X + 1

next X until X = N + 1    endwhile

(1 mark for correct first line of loop construct)

(1 mark for correct loop control and last line of loop construct) [2]

Q4)

(a) 120

1 [2]

(b) for X = 1 to N + 1    OR repeat OR while X <> N + 1 do

(T = T \* X)    (T = T \* X)    (T = T \* X)

X = X + 1    X = X + 1

next X until X = N + 1    endwhile

(1 mark for correct first line of loop construct)

(1 mark for correct loop control and last line of loop construct) [2]

Q5)

Expected output:

1

2

Error

Q6)

(a)

count	number	total	x	average	OUTPUT
1		0	0		
2	15	15	1		
3	-2				
4	0				
5	8	23	2		
6	0				
7	21	44	3		
8	-8				
9	-12				
10	1	45	4		
11	25	70	5	14	14

} 1  
 } 1  
 } 1  
 } 1

<- - - - - 1 mark - - - - -><- 1 mark -><- 1 mark -><- - - - - - 1 mark - - - - -> [4]

(b) Find the average of all positive numbers entered [1]

# **PSEUDOCODE WORKSHEETS**



Q1) Award one mark for each correct step in the algorithm:

- initialise variables/arrays
- loop each student
- input ID
- input log on
- input log off
- calculate difference
- store difference
- calculate average
- output average

MAX 6

[6]

Q2) (a) Award one mark each

- (i) 33.8 [1]
- (ii) 41 [1]

(b) Award one mark for each correct step in the algorithm

- Initialise
- Loop
- Input temperature (x24)
- Convert to Fahrenheit
- Find maximum and minimum
- Calculate average (outside loop)
- Output maximum, minimum, average [5]

Examples of correct answers are:

- sum = 0
- min = 100
- max = 0
- count = 1
- while count <= 24 do
- input temp
- $F = (\text{temp} * 1.8) + 32$
- sum = sum + F
- if  $F < \text{min}$  then min = F
- if  $F > \text{max}$  then max = F
- count = count + 1

- endwhile
  - average = sum/24
  - print average, min, max
- 
- (ii) sum = 0
  - min = 100
  - max = 0
  - count = 1
  - repeat
  - input temp
  - $F = (\text{temp} * 1.8) + 32$
  - sum = sum + F
  - if  $F < \text{min}$  then min = F
  - if  $F > \text{max}$  then max = F
  - count = count + 1
  - until count > 24
  - average = sum/24
  - print average, min, max

Q3) (a) Award 1 mark each for trace and reason:

trace – 3,5,7,9,11.....

reason – x is odd/loop does not terminate/goes on forever [2]

(b) Award 1 mark for the following stages:

- initialise
- loop
- use of  $x = x + 2$
- output of x [3]

Q4) Award 1 mark for each correct step in the algorithm:

Initialise

- Loop
- Input marks (x25)
- Match mark to grade (If..Then..Else or Case ) one correct
- Increment grade total
- Output the number of distinction, merit, pass and fail grades given [6]

Q5) (a) Award one mark

20 [1]

(b) Award one mark for each correct step in the algorithm

- Initialise one mark
- Loop (30) one mark
- Input ID, weight, height one mark
- IF.....THEN.....ELSE three marks
- (or CASE OF.....OTHERWISE)
- Calculate BMI one mark
- Output ID, BMI and comment one mark [6]

Q6) General marking points:

- loop – 1 mark
- input in correct place – 1 mark
- checks on code – 1 mark
- correct use of if/then/else or case statements – 1 mark
- increment all totals – 1 mark
- error recognition/validation – 1 mark
- correct output in correct place – 1 mark [5]

Sample program 1:

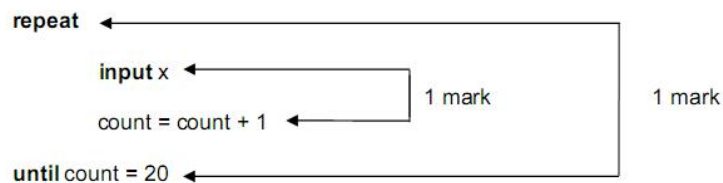
- set c, d, v, b = 0: set count = 0
- repeat 1 mark
- input code 1 mark
- $x = \text{code}/10000$  }
- $y = \text{INT}(x)$  } 1 mark
- if  $y = 1$  then  $c = c + 1$  }
- else if  $y = 2$  then  $d = d + 1$  }
- else if  $y = 3$  then  $v = v + 1$  } 2 marks
- else if  $y = 4$  then  $b = b + 1$  }
- else print "error" 1 mark
- count = count + 1
- until count = 5000
- print c, d, v, b 1 mark

Sample program 2:

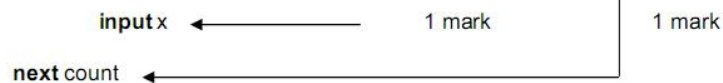
- set c, d, v, b = 0: set count = 0
- repeat 1 mark

- input code 1 mark
  - if code  $\geq 1000$  and code  $< 2000$  then  $c = c + 1$  }
  - else if code  $\geq 2000$  and code  $< 3000$  then  $d = d + 1$  }
  - else if code  $\geq 3000$  and code  $< 4000$  then  $y = y + 1$  } 3 marks
  - else if code  $\geq 4000$  and code  $< 5000$  then  $b = b + 1$  }
  - else print "error" 1 mark
  - count = count + 1
  - until count = 5000
  - print c, d, v, b 1 mark
- (NOTE – OK to use statements such as if code begins with a 1 as code checks)

(a) (i) count = 0



(ii) for count = 1 to 20



[4]

Q7)

(b) while...do

Q8) Sample algorithm:

- input amount
- if amount  $>$  balance then  $x = 1$  (2 marks)
- else if amount  $>$  daily limit then  $x = 1$  (1 mark)
- else  $x = 0$
- while  $x = 0$
- if balance  $< 100$  then charge =  $0.02 * \text{amount}$  (1 mark)
- else charge = 0
- (1 mark)
- endwhile
- if  $x = 1$  then print "Sorry, withdrawal refused"
- print charge (1 mark)

### Marking points

- 1 mark for checking if amount > balance
- 1 mark for checking if amount > daily limit
- 1 mark for some way of testing if withdrawal will be refused (value of x in above)
- 1 mark for checking if balance < \$100...
- 1 mark ...for calculating 2% charge
- 1 mark for no charge if balance >= \$100
- 2 marks for giving correct outputs [5]

Q9) marking points (1 mark per item up to the maximum of 5):

- initialise fa, sj and ka to zero
- correct loop
- inputs (in correct place)
- addition of number of flights per airline
- any validation checks carried out
- calculate percentages
- outputs (in correct place and ONLY if some evidence of any attempt at processing)

sample program/algorithm:

- fa = 0; sj = 0; ka = 0; } 1 mark
- for x = 1 to 400 } 1 mark
- input lettercode }
- } 1 mark
- input numbercode }
- if lettercode = "FA" then fa = fa + 1 }
- }
- if lettercode = "SJ" then sj = sj + 1 } 1 mark
- }
- if lettercode = "KA" then ka = ka + 1 }
- else print "error" } 1 mark
- next x
- fapercent = fa/4 }
- }
- sjpercent = sj/4 } 1 mark
- }
- kapercent = ka/4 }
- print fapercent, sjpercent, kapercent } 1 mark [5]

Q10) Marking points (maximum of 7 marks)

- initialising highest and lowest to reasonable values (must not be zero)
- first loop controlling one year (365 days)
- re-setting total for each day
- second loop controlling readings taken per day
  - read temperature
  - calculate total day temperature
  - calculate total year temperature
  - identifying highest temperature
  - identifying lowest temperature
  - finding average temperature for day
  - finding average temperature for year
- output average day temperature inside loop
- output highest, lowest, average outside the loop

#### Sample algorithm in pseudocode

- highest = -100: lowest = 100: total\_year = 0 } 1 mark
- for c = 1 to 365 } 1 mark
- total\_day = 0 } 1 mark
- for d = 1 to 10 } 1 mark
- read temp } 1 mark
- total\_day = total\_day + temp } mark
- total\_year = total\_year + temp } 1 mark
- if temp > highest then highest = temp } 1 mark
- if temp < lowest then lowest = temp } 1 mark
- next d
- average\_day = total\_day/10 } 1 mark
- print average\_day } 1 mark
- next c
- average\_year = total\_year/365 } 1 mark
- print highest, lowest, average\_year } 1 mark [7]

Q11) (a) total = 0 (1 mark) initialisation

- for x = 1 to 50 (1 mark) correct loop
- input number (1 mark) correct input and output
- if number > 100 then total = total + 1 (1 mark)
- count numbers > 100
- next x
- output total

(1 mark for initialising total)

(1 mark for correct loop – accept repeat loop or a while loop)

(1 mark for correct input (within loop) and output (after the loop))

(1 mark for counting how many input numbers were > 100) [3]

- (b) total = 0 (1 mark) initialise total
- for x = 1 to 100 (1 mark) correct loop
- input number (1 mark) correct input and output
- total = total + number (1 mark) finding sum of numbers
- next x
- average = total/100 (1 mark) calculate average
- output average
- (1 mark for initialising total)
- (1 mark for correct loop – accept repeat loop or a while loop)
- (1 mark for correct input (inside the loop) and output (after the loop))
- (1 mark for calculating total)
- (1 mark for calculating the average outside the loop) [3]

Q12)

(a) input name\$

- input H, M
- if name\$ = "Mexico" then H = H - 7
- else if name\$ = "India" then H = H + 4: M = M + 30
- else if name\$ = "New Zealand" then H = H + 11
- else print "error"
- print H, M

Marking points

- 1 mark for two inputs for country and hours/mins
- 1 mark for check on Mexico
- 1 mark for check on New Zealand
- 1 mark for check on India
- 1 mark for error check
- 1 mark for output in correct place [4]

(b) Any two sets of test data from:

- Normal hours: (hours which do not change the day) e.g. 8
- hours which change the day (e.g.. 13 + country = New Zealand)
- Normal minutes (which do not change the hour) eg.25
- minutes which change the hour (e.g. 40 + country=India) [2]

Q13) marking points

- Initialisation (smallest, largest, total) (could be first input)
- correct loop (also: repeat .... until n = 500, while n <> 500 do ....)
- input (inside a loop)
- calculate the density
- check on largest density + action taken
- check on smallest density + action taken
- find population total + calculate average population
- print values (outside loop + some evidence of processing taking place)

e.g.

- smallest = 10000: largest = 0: total = 0 (1 mark)
- for country = 1 to 500 (1 mark)
- input population, area (1 mark)
- density = population/area (1 mark)
- if density > largest then largest = density (1 mark)
- if density < smallest then smallest = density (1 mark)
- total = total + population
- next country (1 mark)
- average = total/500
- print largest, smallest, average (1 mark) [6]

Q14)



(a) **One mark per correct output:**

Input	Output
0	<b>Fail Grade</b>
5	<b>Pass Grade</b>
99	<b>Not a valid mark</b>

(NOTE: accept the words FAIL, PASS without the word GRADE. If the word GRADE precedes the words FAIL, PASS still accept the answer. The letters "P" and "F" on their own = 0 marks)

[3]

(b) For example (1 mark per line up to the maximum):  
mark = mark/10  
mark = INT(mark)

mark = mark DIV10 is worth 2 marks on its own

[2]

Q15)

Any **five** of the following stages:

INPUT employee	(1) }
	} maximum of
INPUT no_of_years	(1) }
	} 2 marks for
INPUT type_of_good	(1) }
	} all the inputs
INPUT price	(1) }
(if employee <> "yes") <u>then</u> (discount = 0%)	(2)
<u>else</u> (if no_of_years < 5) <u>or</u> (type_of_good = "electrical")	(2)
<u>then</u> discount = 10%	(1)
<u>else</u> discount = 20%	(1)

**up to maximum mark of [5]**

Q16)

- (c) **Repeat ... Until**  
Any **two** points from:

all instructions are carried out within the loop  
code within loop could be executed several times  
testing of loop is done at the end

- If ... then ... else ... endif**  
Any **two** points from:

only the relevant code is executed  
result of test allows only one execution to be done  
testing is done throughout the loop

**NOTE:** Examples on their own gain no marks – a description of both types of programming construct is required [4]

Q19)

(a)

- 150 abnormal reading
- 400 normal speed
- 800 high speed
- (ignore word “speed” in answer) [3]

(b) any two points from:

- only data 0 to 9 would register
- all other data would give “abnormal reading” message/incorrect response
- variable whole would not exist
- thus whole would be zero OR algorithm would crash/fail [2]

Q20)

(a)

- wrong = 0 (1 mark)
- for count = 1 to 50 (1 mark)
- input number (1 mark)
- if number < 1000 or number > 9999 (2 marks)
- then wrong = wrong + 1 (1 mark)
- endif
- next count
- percent = wrong \* 2 (1 mark)

- output wrong, percent (1 mark)
- (accept flow charts but not essays) [6]

(General answer:

- Initialise variables – 1 mark
- Loop control – 1 mark
- Input number – 1 mark
- Check numbers in range – 2 marks
- Increment incorrect numbers total – 1 mark
- Calculate the percentage – 1 mark
- Output totals – 1 mark)

(b) any two validation checks with examples:

- length check
- example: make sure there are always 4 digits/characters input
- character check
- example: make sure only numbers are input and not letters
- type check
- example: 0 decimal places/integer value
- (format check, check digit, presence check = 0)
- (example must tie up with validation check for second mark and conversely) [4]

Q21) Sample answer:

- repeat
- input start\_point }
- input end\_point } 1 mark
- input number }
- cost = abs(start\_point - end\_point) \* number \* 2 } 2 marks
- if number >= 3 then cost = cost – (cost/10) } 1 mark
- input money } 1 mark
- change = money – cost } 1 mark
- for x = 1 to number }
- print ticket } 1 mark
- next x } 1 mark
- output change }

- until no more customers } 1 mark

IMRAN KHAN

General marking points:

- (initialisation = 0)
- inputs – 1 mark
- calculate how many stations to charge for – 1 mark
- formula/if statement to calculate cost for ticket/no discount - 1 mark
- formula/if statement to calculate discount where appropriate - 1 mark
- input money - 1 mark
- formula to calculate change - 1 mark
- loop to control number of tickets to be printed - 1 mark
- print ticket/output change - 1 mark
- overall loop control - 1 mark [6]

Q22) Sample program

- m1 = 100 }
- m2 = 0 } 1 mark
- sum = 0 }
- n = 1 }
- while n < 151 do 1 mark
- repeat
- read mark 1 mark
- until (mark >= 0 and) mark < 101 1 mark (validation check)
- if mark < m1 then m1 = mark 1 mark
- if mark > m2 then m2 = mark 1 mark
- sum = sum + mark 1 mark
- n = n + 1
- endwhile
- average = sum/150 1 mark
- output average, m1, m2 1 mark [6]

General mark points

- initialisation (must correctly set smallest (m1) and largest (m2) boundaries)
- method for looping round for 150 students
- reading in marks for all students
- checking if mark inside 0 to 100 boundary and action taken
- setting value of smallest (m1) after checking against input mark
- setting value of largest (m2) after checking against input mark

- totalling all marks together
- calculating the average mark
- output of average, smallest mark (m1), largest mark (m2)

Q23)(a)

- error 1: product = 0 on line 2
- should use product = 1
- error 2: loop control, count <= 10 on line 3
- should use count < 10 or alternatively alter count value on line 1 to count = 1
- error 3: print value of product inside loop on line 7
- output should come after the endwhile statement [3]

(b) Accept either of the following loop controls:

- repeat for count = 1 to 10
- OR
- until count = 10 next count
- (accept repeat
- until count > 11
- if line 1 changed to count = 1) [1]

Q24) Sample program:

- count = 0
- total1 = 0
- total2 = 0
- lowest = 1000 1 mark
- while count < 200 do 1 mark
- input temp 1 mark
- if temp < 10 then total1 = total1+1 1 mark
- if temp > 20 then total2 = total2+1 1 mark
- if temp < lowest then lowest = temp 1 mark
- count = count + 1
- endwhile
- output total1, total2, lowest 1 mark

(max of 5 marks)

Marking points:

- Initialisation (but lowest must be set to a suitable value)
  - Correct loop to read in 200 temperatures
  - Correct input for temperatures
  - Check if temperature is less than 10 and increment total1
  - Check if temperature greater than 20 and increment total2
  - Identifying the lowest temperature
  - Output results (only give output mark if some data processing has been done, and outside loop)
- [5]

Q25) (a)  $40/10 = 4$  [1]

(b) general marking points

- initialising best and worst to sensible values
- correct loop for 1000 cars
- correct use of calculation given in part (a)
- output economy for each car inside loop
- determining best economy
- determining worst economy
- calculating mean economy for all cars
- input data and output all three results (only award mark if some form of processing done) [6]

sample program

- total = 0, count = 0, best = 0, worst = 1000 1 mark
- repeat 1 mark
- input litres, distance
- economy = distance/litres 1 mark
- print economy 1 mark
- if economy > best then best = economy 1 mark
- if economy < worst then worst = economy 1 mark
- total = total + economy
- count = count + 1
- until count = 1000
- average = total/1000 1 mark
- print average, best, worst 1 mark

Q26) Marking points

- correct loop
- correct inputs

- check for type and calculate itemcost
- action taken if type NOT 1, 2 or 3
- calculate totalcost
- calculate the average totalcost
- both outputs in the correct place

Sample algorithm:

- total cost = 0
- for x = 1 to 1000 (1 mark)
- input type, partcost (1 mark)
- if type = 1 then itemcost = partcost \* 1.5}
- if type = 2 then itemcost = partcost \* 2.5} (1 mark)
- if type = 3 then itemcost = partcost \* 5.0}
- else print error (1 mark)
- totalcost = totalcost + itemcost (1 mark)
- print itemcost
- next x
- average = totalcost/1000 (1 mark)
- print average (1 mark) [5]

Q27)

(a) 100 (km/hr) [1]

(b) Marking points

- Initialisation (slowest = 1000 or an equivalent high value)
- Correct loops structure and control
- Input (in correct place)
- Calculation of final speed using given formula in part (a) inside the loop
- Output the final speed for ALL cars inside the loop
- Calculation highest speed input
- Calculation slowest speed input
- Calculate the average (two parts to this calculation)
- Final outputs (correct place + some form of processing done) [6]

Sample program:

- total = 0 }
- highest = 0 } 1 mark
- slowest = 1000 }



- for n = 1 to 500 } 1 mark
- input time } 1 mark
- finalspeed = 200/time } 1 mark
- print finalspeed } 1 mark
- total = total + finalspeed
- if finalspeed > highest }
- then highest = finalspeed } 1 mark
- if finalspeed < slowest }
- then slowest = finalspeed } 1 mark
- next n
- average = total/500 } 1 mark
- print average, highest, slowest } 1 mark

IMRAN KHAN

Q28) mark for each error and 1 mark for reason why it is an error

- line 1/negative=1 and/or line 2/positive=1
- negative and/or positive should be set to zero
- line 7/count=count+1
- don't need a count within a for .... to next loop
- replace loop with a repeat...until loop
- line 8/print negative, positive or line 9/next count
  - - outputs should come after the next count statement [6]

Q29) Marking Points

- initialisation of running totals (1 mark)
- correct loop control (1 mark)
- error trap for height input (1 mark)
- error trap for weight input (1 mark)
- sum total1 and average1 (i.e. height) calculation (1 mark)
- sum total2 and average2 (i.e. weight) calculation (1 mark)
- correct output (only if some processing attempted, must be outside loop) (1 mark)
- [max: 5]

Sample pseudocode

- total1 = 0: total2 = 0 (1 mark)
- for x = 1 to 1000 (1 mark)
- input height, weight
- if height > 2 or height < 0 then print "error": input height (1 mark)
- if weight > 130 or weight < 0 then print "error": input weight (1 mark)
- else total1 = total1 + height: total2 = total2 + weight
- next x
- average1 = total1/1000 (1 mark)
- average2 = total2/1000 (1 mark)
- print average1, average2 (1 mark) [5]

Q30)

(a) marking points:

the way to find and print the largest value a 1 mark

the way to find and print the largest value b 1 mark

the way to find and print the largest value c 1 mark

sample algorithm:

input a, b, c

if a > b and a > c then print a (1 mark)

else if  $b > c$  then print b (1 mark)  
else print c (1 mark) [3]

(b) marking points:

loop construct 1 mark

check if number is an integer 1 mark

counting the number of integers input 1 mark

output count value (outside the loop) 1 mark

sample algorithm:

for  $x = 1$  to 1000 (1 mark)

input number

difference =  $\text{INT}(\text{number}) - \text{number}$  (1 mark)

if difference = 0 then total = total + 1 (1 mark)

next x

print total (1 mark)

(NOTE: alternative to lines 3 and 4:

if  $\text{INT}(\text{number}) = \text{number}$  then total = total + 1 (2 marks) ) [4]

Q31) Marking points

initialise variables 1 mark

correct loop control 1 mark

input (in correct place) 1 mark

correct check on type of weather (if, case, etc.) 1 mark

adding number of days of each type of weather 1 mark

check for the highest temperature 1 mark

check for the lowest temperature 1 mark

output (all items in the correct place) 1 mark

Sample algorithm

$c = 0$ ;  $r = 0$ ;  $s = 0$ ;  $f = 0$

high = 0 (or a negative number)

low = 1000 (1 mark)

for  $x = 1$  to 365 (1 mark)

input weather, temp (1 mark)

if weather = "CLOUDY" then  $c = c + 1$

else if weather = "RAINING" then  $r = r + 1$  (2 marks)

else if weather = "SUNNY" then  $s = s + 1$

else if weather = "FOGGY" then  $f = f + 1$

endif

if temp > high then high = temp (1 mark)

```
if temp < low then low = temp (1 mark)
next x
print c, r, s, f, high, low (1 mark) [6]
```

IMRAN KHAN

Q32)

(a) sample program:

```
x = 0: y = 0 (1 mark)
input number (1 mark)
while number <> -1 do (1 mark)
if number > 1000 then x = x + 1 (1 mark)
else if number < 1000 then y = y + 1 (1 mark)
input number
endwhile
print x, y (1 mark)
```

marking points:

- initialisation of variables
- first and subsequent inputs in the correct place
- correct loop control (only repeat or while loops work here)
- check if number > 1000 and increment total
- check if number < 1000 and increment total
- output totals outside the loop [4]

(b) sample program

```
T = 0
for N = 1 to 50 (1 mark)
read D1, D2, D3, D4 (1 mark)
if D1 = D4 and D2 = D3 then T = T+1 (2 marks)
next N
percent = T * 2
print percent (1 mark)
```

marking points

- correct loop (for, repeat or while loops all work)
- correct input
- check whether D1 = D4 and D2 = D3
- summation if D1 = D4 and D2 = D3
- calculate percentage and output the value outside the loop [4]