

---

**COMPUTER SCIENCE**

**9608/21**

Paper 2 Written Paper

**May/June 2018**

MARK SCHEME

Maximum Mark: 75

---

**Published**

This mark scheme is published as an aid to teachers and candidates, to indicate the requirements of the examination. It shows the basis on which Examiners were instructed to award marks. It does not indicate the details of the discussions that took place at an Examiners' meeting before marking began, which would have considered the acceptability of alternative answers.

Mark schemes should be read in conjunction with the question paper and the Principal Examiner Report for Teachers.

Cambridge International will not enter into discussions about these mark schemes.

Cambridge International is publishing the mark schemes for the May/June 2018 series for most Cambridge IGCSE™, Cambridge International A and AS Level and Cambridge Pre-U components, and some Cambridge O Level components.

---

IGCSE™ is a registered trademark.

This document consists of **14** printed pages.

**Generic Marking Principles**

These general marking principles must be applied by all examiners when marking candidate answers. They should be applied alongside the specific content of the mark scheme or generic level descriptors for a question. Each question paper and mark scheme will also comply with these marking principles.

**GENERIC MARKING PRINCIPLE 1:**

Marks must be awarded in line with:

- the specific content of the mark scheme or the generic level descriptors for the question
- the specific skills defined in the mark scheme or in the generic level descriptors for the question
- the standard of response required by a candidate as exemplified by the standardisation scripts.

**GENERIC MARKING PRINCIPLE 2:**

Marks awarded are always **whole marks** (not half marks, or other fractions).

**GENERIC MARKING PRINCIPLE 3:**

Marks must be awarded **positively**:

- marks are awarded for correct/valid answers, as defined in the mark scheme. However, credit is given for valid answers which go beyond the scope of the syllabus and mark scheme, referring to your Team Leader as appropriate
- marks are awarded when candidates clearly demonstrate what they know and can do
- marks are not deducted for errors
- marks are not deducted for omissions
- answers should only be judged on the quality of spelling, punctuation and grammar when these features are specifically assessed by the question as indicated by the mark scheme. The meaning, however, should be unambiguous.

**GENERIC MARKING PRINCIPLE 4:**

Rules must be applied consistently e.g. in situations where candidates have not followed instructions or in the application of generic level descriptors.

**GENERIC MARKING PRINCIPLE 5:**

Marks should be awarded using the full range of marks defined in the mark scheme for the question (however; the use of the full mark range may be limited according to the quality of the candidate responses seen).

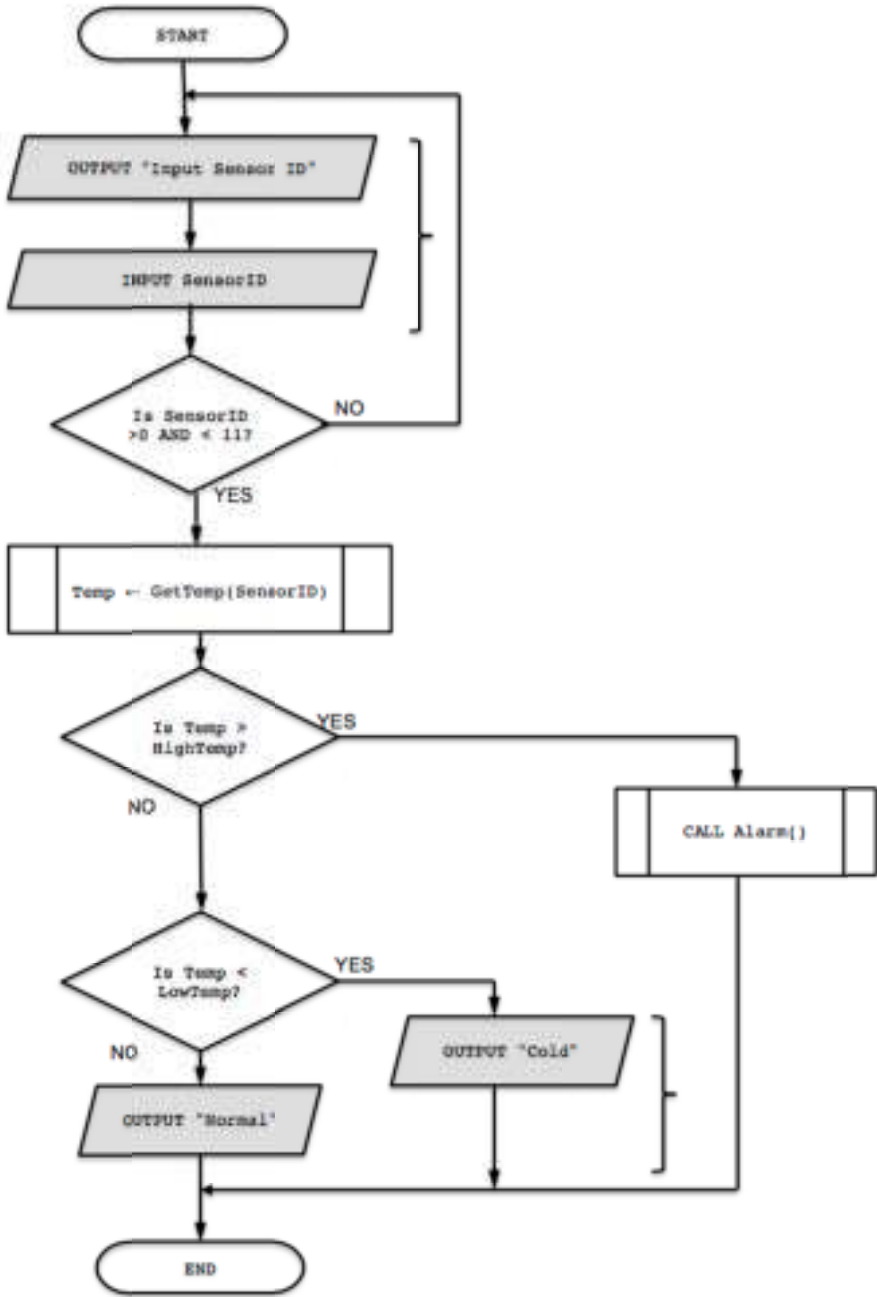
**GENERIC MARKING PRINCIPLE 6:**

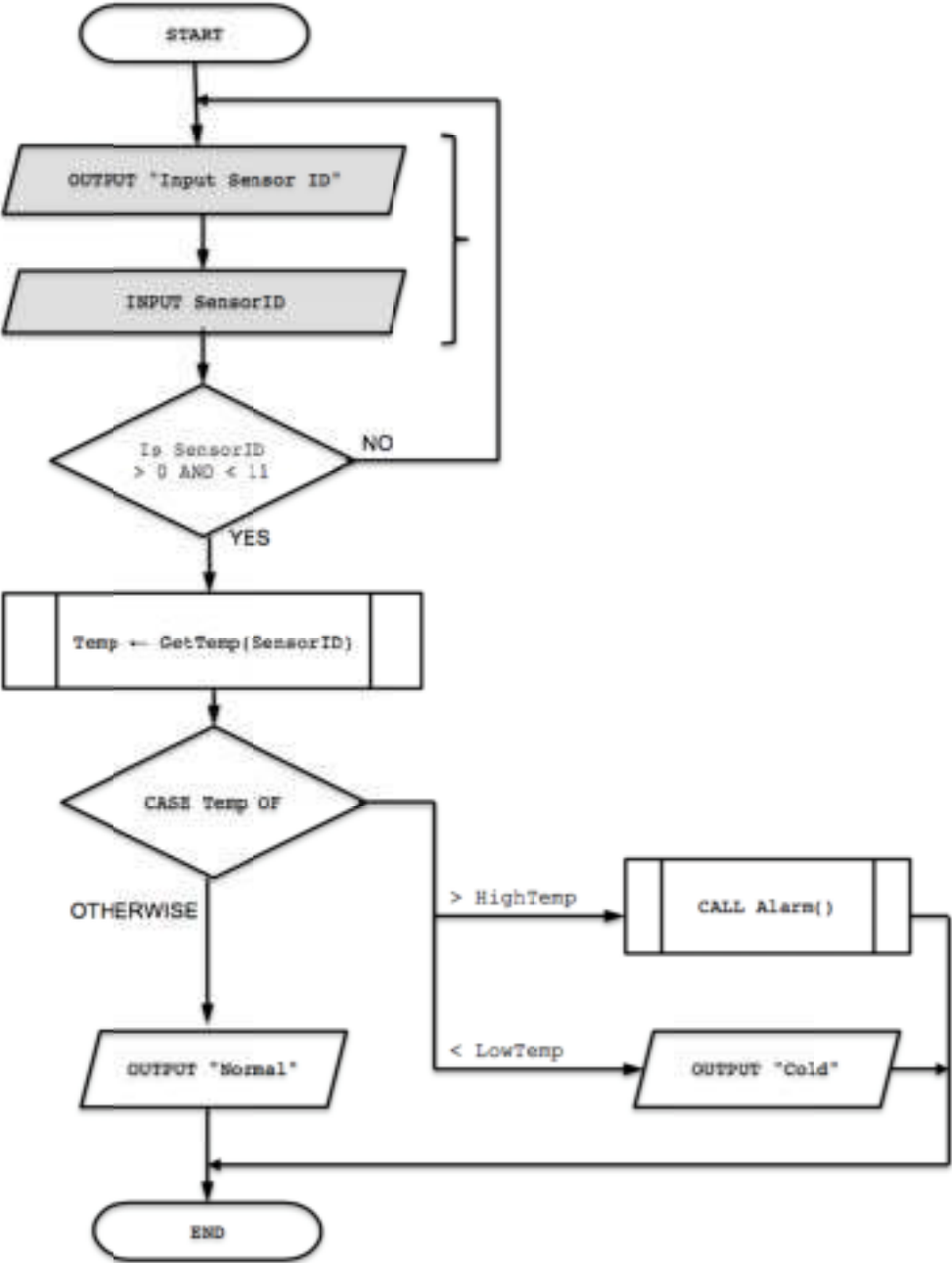
Marks awarded are based solely on the requirements as defined in the mark scheme. Marks should not be awarded with grade thresholds or grade descriptors in mind.

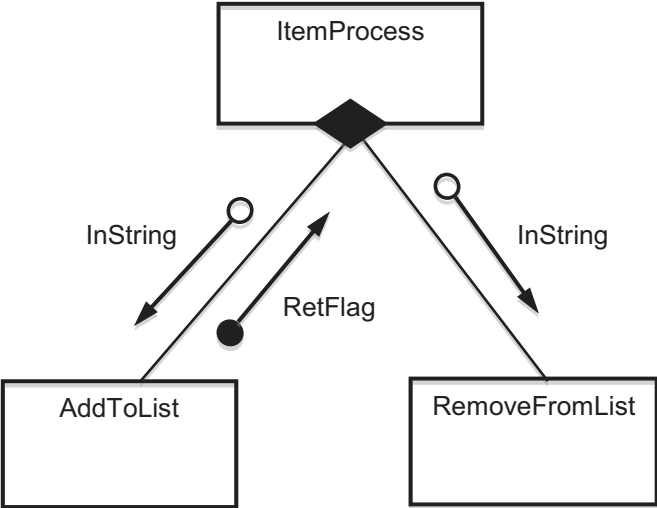
Question	Answer	Marks												
1(a)	<table border="1"> <thead> <tr> <th>Description of Data Item</th> <th>Suitable Identifier Name</th> </tr> </thead> <tbody> <tr> <td>The temperature of the patient</td> <td>PatientTemperature</td> </tr> <tr> <td>The temperature of the room</td> <td>RoomTemperature</td> </tr> <tr> <td>The patient identification number</td> <td>PatientID</td> </tr> <tr> <td>The name of the nurse taking the measurement</td> <td>NurseName</td> </tr> </tbody> </table> <p>The above are examples only. Names must be meaningful and unambiguous</p> <p>Items 1 and 2 must have suitable prefix / suffix (i.e. not just 'temp')</p>	Description of Data Item	Suitable Identifier Name	The temperature of the patient	PatientTemperature	The temperature of the room	RoomTemperature	The patient identification number	PatientID	The name of the nurse taking the measurement	NurseName	4		
Description of Data Item	Suitable Identifier Name													
The temperature of the patient	PatientTemperature													
The temperature of the room	RoomTemperature													
The patient identification number	PatientID													
The name of the nurse taking the measurement	NurseName													
1(b)(i)	<table border="1"> <thead> <tr> <th>Expression</th> <th>Evaluates to</th> </tr> </thead> <tbody> <tr> <td>"Mon" &amp; MID(MyGreeting, 10, 2)</td> <td>"Month"</td> </tr> <tr> <td>AgeInYears + ASC(MyInitial)</td> <td>94</td> </tr> <tr> <td>INT(MyInitial)</td> <td>ERROR</td> </tr> <tr> <td>MOD(Weight * 2, 10)</td> <td>1</td> </tr> <tr> <td>Married AND (NOT Children)</td> <td>FALSE</td> </tr> </tbody> </table>	Expression	Evaluates to	"Mon" & MID(MyGreeting, 10, 2)	"Month"	AgeInYears + ASC(MyInitial)	94	INT(MyInitial)	ERROR	MOD(Weight * 2, 10)	1	Married AND (NOT Children)	FALSE	5
Expression	Evaluates to													
"Mon" & MID(MyGreeting, 10, 2)	"Month"													
AgeInYears + ASC(MyInitial)	94													
INT(MyInitial)	ERROR													
MOD(Weight * 2, 10)	1													
Married AND (NOT Children)	FALSE													
1(b)(ii)	<table border="1"> <thead> <tr> <th>Variable</th> <th>Data type</th> </tr> </thead> <tbody> <tr> <td>MyGreeting</td> <td>STRING</td> </tr> <tr> <td>MyInitial</td> <td>CHAR</td> </tr> <tr> <td>AgeInYears</td> <td>INTEGER</td> </tr> <tr> <td>Weight</td> <td>REAL</td> </tr> <tr> <td>Married</td> <td>BOOLEAN</td> </tr> </tbody> </table> <p>One mark per answer</p> <p>Alternative appropriate data types acceptable</p>	Variable	Data type	MyGreeting	STRING	MyInitial	CHAR	AgeInYears	INTEGER	Weight	REAL	Married	BOOLEAN	5
Variable	Data type													
MyGreeting	STRING													
MyInitial	CHAR													
AgeInYears	INTEGER													
Weight	REAL													
Married	BOOLEAN													

Question	Answer	Marks
2(a)(i)	<ul style="list-style-type: none"> <li>• Indentation</li> <li>• Blank lines / white space</li> <li>• Capitalisation of keywords</li> <li>• Meaningful identifier names</li> </ul>	4
2(a)(ii)	<ul style="list-style-type: none"> <li>• Comments</li> </ul>	1

Question	Answer		Marks
2(b)	<b>Feature</b>		<b>9</b>
	<b>Answer</b>		
	A line containing an example of an assignment statement	08,12,13,19	
	A line containing the start of a repetition block	10	
	A line containing the end of a repetition block	23	
	The line containing the start of a selection statement	15	
	The number of parameters of the MID function	3	
	The boolean operator used	AND	
	The number of local variables	4	
	The number of function calls from within <code>StringClean()</code> resulting from the call: <code>NewString ← StringClean("Me")</code>	5	
The number of a line containing an unnecessary statement	06		

Question	Answer	Marks
3	 <pre> graph TD     Start([START]) --&gt; Output1[/OUTPUT "Input Sensor ID"/]     Output1 --&gt; Input[/INPUT SensorID/]     Input --&gt; Decision1{Is SensorID &gt; 0 AND &lt; 11?}     Decision1 -- NO --&gt; Input     Decision1 -- YES --&gt; Process1[Temp ← GetTemp(SensorID)]     Process1 --&gt; Decision2{Is Temp &gt; HighTemp?}     Decision2 -- YES --&gt; Process2[CALL Alarm()]     Decision2 -- NO --&gt; Decision3{Is Temp &lt; LowTemp?}     Decision3 -- YES --&gt; Output2[/OUTPUT "Cold"/]     Decision3 -- NO --&gt; Output3[/OUTPUT "Normal"/]     Output2 --&gt; End([END])     Output3 --&gt; End     Process2 --&gt; End   </pre> <p>One mark for:</p> <ol style="list-style-type: none"> <li>1 START and END / STOP</li> <li>2 prompt and input of SensorID (allow alt name)</li> <li>3 decision box checking that SensorID is between 1 &amp; 10</li> <li>4 calling GetTemp with SensorID as parameter</li> <li>5 decision box comparing Temp &gt; HighTemp</li> <li>6 calling Alarm()</li> <li>7 decision box comparing Temp &lt; LowTemp</li> <li>8 both output messages</li> </ol>	8

Question	Answer	Marks
3	<p>ALTERNATIVE SOLUTION USING 'CASE'</p>  <pre> graph TD     Start([START]) --&gt; Output1[/OUTPUT "Input Sensor ID"/]     Output1 --&gt; Input[/INPUT SensorID/]     Input --&gt; Decision1{Is SensorID &gt; 0 AND &lt; 11}     Decision1 -- NO --&gt; Output1     Decision1 -- YES --&gt; Process1[Temp ← GetTemp(SensorID)]     Process1 --&gt; Decision2{CASE Temp OF}     Decision2 -- OTHERWISE --&gt; Output2[/OUTPUT "Normal"/]     Decision2 -- "&gt; HighTemp" --&gt; Process2[CALL Alarm()]     Decision2 -- "&lt; LowTemp" --&gt; Output3[/OUTPUT "Cold"/]     Output2 --&gt; End([END])     Process2 --&gt; End     Output3 --&gt; End   </pre> <p>Mark as follows:</p> <ol style="list-style-type: none"> <li>1 One mark for START and END / STOP</li> <li>2 One mark for prompt and input of SensorID (allow alt name)</li> <li>3 One mark for decision box checking that SensorID is between 1 &amp; 10</li> <li>4 One mark for calling GetTemp with SensorID as parameter</li> <li>5 One mark for decision box CASE Temp</li> <li>6 One mark for calling Alarm()</li> <li>7 One mark for two correct CASE conditions</li> <li>8 One mark for both output messages</li> </ol>	8

Question	Answer	Marks
4(a)	<p>Features:</p> <ul style="list-style-type: none"> <li>• The <u>hierarchy</u> of modules</li> <li>• The <u>parameters</u> that are passed between modules // the <u>interface</u> between the modules</li> <li>• The <u>sequence</u> of module execution</li> </ul> <p>One mark per item</p>	<b>3</b>
4(b)	 <p>Mark as follows:            One mark for top box            One mark for <b>both</b> lower boxes            One mark for diamond 'decision' symbol            One mark for each parameter (3 parameters)</p>	<b>6</b>

Question	Answer	Marks
5(a)	Details are saved after the program ends // after the computer is switched off	<b>1</b>
5(b)	<p>Two from the following examples:</p> <ul style="list-style-type: none"> <li>• Context-sensitive help</li> <li>• Syntax checking (on entry)</li> <li>• Automatic indentation</li> <li>• Type checking (Parameter checking)</li> <li>• PrettyPrinting</li> <li>• Highlight structure blocks (e.g. selection, iteration)</li> <li>• Highlight any undeclared variables</li> <li>• Highlight any unassigned variables</li> </ul>	<b>Max 2</b>

Question	Answer	Marks
5(c)	<p>'Pseudocode' solution included here for development and clarification of mark scheme. Programming language solutions appear in the Appendix.</p> <pre> PROCEDURE AddNewScores()    DECLARE FileData : STRING   DECLARE ScoreDate : STRING   DECLARE MembershipNumber : STRING   DECLARE Score : STRING    OUTPUT "Input the date for the scores"   INPUT ScoreDate    OPENFILE "ScoreDetails.txt" FOR APPEND    OUTPUT "Input the Membership number"   INPUT MembershipNumber    WHILE NOT MembershipNumber = ""     OUPUT "Input the score"     INPUT Score     WHILE (INT(SCORE) &lt; 50) OR (INT(SCORE) &gt; 99)       OUTPUT "Input a valid score from 50 to 99"       INPUT Score     ENDWHILE     FileData = MembershipNumber &amp; ScoreDate &amp; Score     WRITEFILE "ScoreDetails.txt", FileData     OUTPUT "Input the Membership number"     INPUT MembershipNumber    ENDWHILE    CLOSEFILE("ScoreDetails.txt")  ENDPROCEDURE </pre> <p>1 mark for each of the following:</p> <ol style="list-style-type: none"> <li>1 <b>Declare</b> MembershipNumber <b>as</b> STRING <b>and</b> Score <b>as</b> INTEGER / STRING (commented in Python)</li> <li>2 <b>Prompt and Input of</b> ScoreDate</li> <li>3 <b>Open</b> ScoreDetails.txt <b>in</b> APPEND mode</li> <li>4 <b>Prompt and Input of</b> MembershipNumber <b>in a loop</b></li> <li>5 (Outer) loop <b>terminated when</b> MembershipNumber = ""</li> <li>6 <b>Input</b> Score <b>and</b> loop until valid</li> <li>7 <b>Form</b> the text string from the three variables</li> <li>8 <b>Write</b> the text string to the file</li> <li>9 <b>Close</b> the file</li> </ol>	9



Question	Answer	Marks
6(a)(i)	<ul style="list-style-type: none"> <li>• The array is 1D</li> <li>• 1 is the lower bound</li> <li>• 5 is the upper bound</li> <li>• size of array / number of elements = 5</li> </ul>	<b>Max2</b>
6(a)(ii)	<ul style="list-style-type: none"> <li>• subscript / index</li> </ul>	<b>1</b>
6(b)	<pre> FUNCTION Lighten() RETURNS BOOLEAN   DECLARE OldPixelValue : INTEGER   DECLARE NewPixelValue : INTEGER   DECLARE PixelTemp : REAL   DECLARE BurnFlag : BOOLEAN   DECLARE i : INTEGER   DECLARE j : INTEGER    BurnFlag ← FALSE    FOR i ← 1 TO 8     FOR j ← 1 TO 8       OldPixelValue ← Picture[i, j]       PixelTemp ← OldPixelValue * 1.1       NewPixelValue ← INT(PixelTemp)       IF NewPixelValue &gt;= 255         THEN           NewPixelValue ← 255           BurnFlag ← TRUE         ENDIF       Picture[i, j] ← NewPixelValue     ENDFOR   ENDFOR    RETURN BurnFlag  ENDFUNCTION </pre> <p>1 mark for each of the following:</p> <ol style="list-style-type: none"> <li>1 Function heading as above and ending</li> <li>2 Declare <b>and</b> initialise local variable for return <code>BOOLEAN</code> / other mechanism to record 'burnt out'</li> <li>3 Declare local variables for loop counters</li> <li>4 Correct nested loops</li> <li>5 Accessing element from array</li> <li>6 Calculating new value <b>and</b> convert to an <code>INTEGER</code></li> <li>7 Comparing new value with 255 and if greater:</li> <li>8 ...limit to 255 and assign to original element</li> <li>9 ...Set flag / other mechanism if limit applied (Only change once)</li> <li>10 Return a <code>BOOLEAN</code> (following conversion if other mechanism used) <b>MUST WORK</b></li> </ol>	<b>MAX8</b>

Question	Answer	Marks
7	<p>'Pseudocode' solution included here for development and clarification of mark scheme. Programming language solutions appear in the Appendix.</p> <pre> FUNCTION ProcessMarks(Mark: ARRAY[1:20] OF INTEGER) RETURNS INTEGER   DECLARE Highest : INTEGER   DECLARE Average as REAL   DECLARE Total as INTEGER   DECLARE Position as INTEGER    Total ← 0   Highest ← Mark[1] //The highest mark is the first one   Position ← 1    FOR i ← 1 to 20     Total ← Total + Mark[i]     IF Mark[i] &gt; Highest       THEN         Highest ← Mark[i]         Position ← i     ENDFIF   ENDFOR   Average ← Total/20   Output ("The average mark is " &amp; Average &amp; " and the highest mark is " &amp; Highest)   RETURN Position  ENDFUNCTION </pre> <p>1 mark for each of the following:</p> <ol style="list-style-type: none"> <li>1 Correct Function heading (including Mark as parameter) and ending</li> <li>2 Declare local variable for Highest and initialise</li> <li>3 Loop structure (1 to 20 or 0 to 19)</li> <li>4 Comparison with current Highest</li> <li>5 ...Assign new Highest</li> <li>6 Calculate Average</li> <li>7 Output message including both variables and explanatory text</li> <li>8 Return value of index</li> </ol>	Max7

\*\*\* End of Mark Scheme – example program code solutions follow \*\*\*

**Appendix****Program Code Example Solutions****Q5(c)****Visual Basic**

```
Sub AddNewScores()  
  
    Dim FileData As String  
    Dim MembershipNumber As String  
    Dim Score As Integer  
    Dim ScoreDate As String  
    Dim FileHandle As IO.StreamWriter  
  
    FileHandle = New IO.StreamWriter("ScoreDetails.txt")  
  
    Console.WriteLine("Input the date for the scores")  
    ScoreDate = Console.ReadLine()  
  
    Console.WriteLine("Input the Membership number")  
    MembershipNumber = Console.ReadLine()  
  
    Do While MembershipNumber <> ""  
        Console.WriteLine("Input the score")  
        Score = Console.ReadLine()  
        Do While Score < 50 Or Score > 99  
            Console.WriteLine("Input a valid score from 50 to 99")  
            Score = Console.ReadLine()  
        Loop  
        FileData = MembershipNumber & ScoreDate & Str(Score)  
        FileHandle.WriteLine(FileData)  
        Console.WriteLine("Input the Membership number")  
        MembershipNumber = Console.ReadLine()  
    Loop  
    FileHandle.Close()  
  
End Sub
```

**Python**

```
def AddNewScores():
    #ScoreDate as string
    #MembershipNumber as string
    #Score as integer
    #FileHandle as text file

    FileHandle = open("ScoreDetails.txt", "a")
    ScoreDate = str(input("Input the date for the scores"))
    MembershipNumber = str(input("Input the Membership number"))
    while MembershipNumber != "":
        Score = int(input("Input the score"))
        while Score < 50 or Score > 99:
            Score = int(input("Input a valid score from 50 to 99"))
            FileData = MembershipNumber + ScoreDate + str(Score)
            FileHandle.write(FileData)
            MembershipNumber = str(input("Input the Membership number"))
    FileHandle.close
```

**Pascal**

```
procedure AddNewScores;
var FileData, ScoreDate, MembershipNumber: String;
    Score: Integer;
    MyFile: text;

begin
    assign(MyFile, 'ScoreDetails.txt');
    append(MyFile);
    writeln('Input the date for the scores');
    readln(ScoreDate);
    writeln('Input the Membership number');
    readln(MembershipNumber);

    while MembershipNumber <> '' do
        begin
            writeln('Input the score');
            readln(Score);
            while (Score < 50) or (Score > 99) do
                begin
                    writeln('Input the score');
                    readln(Score);
                end;
            FileData := MembershipNumber + ScoreDate + IntToStr(Score);
            write(MyFile, FileData);
            writeln('Input the Membership number');
            readln(MembershipNumber);
        end;
    close (MyFile);
end;
```

**Q7****Visual Basic**

```
Function ProcessMarks(ByVal Mark() As Integer) As Integer

    Dim Highest As Integer
    Dim Average As Single
    Dim Total As Integer
    Dim Position As Integer
    Dim i As Integer

    Total = 0
    Position = 1
    Highest = Mark(1)
    For i = 1 To 20
        Total = Total + Mark(i)
        If Mark(i) > Highest Then
            Highest = Mark(i)
            Position = i
        End If
    Next
    Average = Total / 20
    Console.WriteLine ("The average mark is " & Average & _
        " and the highest mark is " & Highest)

    Return Position
End Function
```

**Python**

```
def ProcessMarks (mark):
    #highest, i, position, total as integer
    #average as real

    highest = mark[0]
    total = 0
    position = 0
    for i in range(0,20):
        total = total + mark[i]
        if mark[i] > highest:
            highest = mark[i]
            position = i
    average = total/20
    print('The average mark is ' + str(average) + \
        ' and the highest mark is ' + str(highest))
    return position
```

**Pascal**

```
function ProcessMarks (mark:array of integer):integer;

var highest, total, position, i: integer;
    average: real;

begin
    highest := mark[1];
    total := 0;
    position := mark[1];
    for i := 1 to 20 do
        begin
            total := total + mark[i];
            if mark[i] > highest then
                begin
                    highest := mark[i];
                    position := i;
                end;
            end;
        average := total / 20;
        writeln ('The average mark is ', average, ' and the highest mark is ',
highest);
        ProcessMarks := position;
    end;
```