

---

**COMPUTER SCIENCE**

**9608/23**

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 **13** 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" data-bbox="347 280 1284 533"> <thead> <tr> <th data-bbox="352 286 847 331">Description of data item</th> <th data-bbox="847 286 1279 331">Suitable identifier name</th> </tr> </thead> <tbody> <tr> <td data-bbox="352 331 847 376">The temperature inside the house</td> <td data-bbox="847 331 1279 376">InsideTemperature</td> </tr> <tr> <td data-bbox="352 376 847 421">The temperature outside the house</td> <td data-bbox="847 376 1279 421">OutsideTemperature</td> </tr> <tr> <td data-bbox="352 421 847 465">The wind speed</td> <td data-bbox="847 421 1279 465">WindSpeed</td> </tr> <tr> <td data-bbox="352 465 847 526">Whether it was raining or not</td> <td data-bbox="847 465 1279 526">WasRaining</td> </tr> </tbody> </table> <p data-bbox="316 566 922 633">The above are examples only. Names must be meaningful and unambiguous</p> <p data-bbox="316 667 1262 701">Items 1 and 2 must have suitable prefix/suffix (i.e. not just 'temperature')</p> <p data-bbox="316 734 659 768">Reject single letter names</p>	Description of data item	Suitable identifier name	The temperature inside the house	InsideTemperature	The temperature outside the house	OutsideTemperature	The wind speed	WindSpeed	Whether it was raining or not	WasRaining	4		
Description of data item	Suitable identifier name													
The temperature inside the house	InsideTemperature													
The temperature outside the house	OutsideTemperature													
The wind speed	WindSpeed													
Whether it was raining or not	WasRaining													
1(b)(i)	<table border="1" data-bbox="368 831 1262 1133"> <thead> <tr> <th data-bbox="373 837 1054 882">Expression</th> <th data-bbox="1054 837 1257 882">Evaluates to</th> </tr> </thead> <tbody> <tr> <td data-bbox="373 882 1054 927">MID(MyName, 4, 4) &amp; "ol"</td> <td data-bbox="1054 882 1257 927">"phenol"</td> </tr> <tr> <td data-bbox="373 927 1054 972">QualityConfirmed AND (Factor &gt;= 6.5)</td> <td data-bbox="1054 927 1257 972">TRUE</td> </tr> <tr> <td data-bbox="373 972 1054 1016">20 + ASC(Quality)</td> <td data-bbox="1054 972 1257 1016">88</td> </tr> <tr> <td data-bbox="373 1016 1054 1061">QualityConfirmed + 3</td> <td data-bbox="1054 1016 1257 1061">ERROR</td> </tr> <tr> <td data-bbox="373 1061 1054 1106">MOD(Factor * 2, 9)</td> <td data-bbox="1054 1061 1257 1106">4</td> </tr> </tbody> </table>	Expression	Evaluates to	MID(MyName, 4, 4) & "ol"	"phenol"	QualityConfirmed AND (Factor >= 6.5)	TRUE	20 + ASC(Quality)	88	QualityConfirmed + 3	ERROR	MOD(Factor * 2, 9)	4	5
Expression	Evaluates to													
MID(MyName, 4, 4) & "ol"	"phenol"													
QualityConfirmed AND (Factor >= 6.5)	TRUE													
20 + ASC(Quality)	88													
QualityConfirmed + 3	ERROR													
MOD(Factor * 2, 9)	4													
1(b)(ii)	<table border="1" data-bbox="440 1196 1190 1498"> <thead> <tr> <th data-bbox="445 1202 762 1247">Variable</th> <th data-bbox="762 1202 1185 1247">Data type</th> </tr> </thead> <tbody> <tr> <td data-bbox="445 1247 762 1292">QualityConfirmed</td> <td data-bbox="762 1247 1185 1292">BOOLEAN</td> </tr> <tr> <td data-bbox="445 1292 762 1337">DayNumber</td> <td data-bbox="762 1292 1185 1337">INTEGER</td> </tr> <tr> <td data-bbox="445 1337 762 1382">Factor</td> <td data-bbox="762 1337 1185 1382">REAL</td> </tr> <tr> <td data-bbox="445 1382 762 1426">Quality</td> <td data-bbox="762 1382 1185 1426">CHAR</td> </tr> <tr> <td data-bbox="445 1426 762 1487">MyName</td> <td data-bbox="762 1426 1185 1487">STRING</td> </tr> </tbody> </table> <p data-bbox="316 1532 603 1565">One mark per answer</p>	Variable	Data type	QualityConfirmed	BOOLEAN	DayNumber	INTEGER	Factor	REAL	Quality	CHAR	MyName	STRING	5
Variable	Data type													
QualityConfirmed	BOOLEAN													
DayNumber	INTEGER													
Factor	REAL													
Quality	CHAR													
MyName	STRING													

Question	Answer	Marks																				
2(a)	Comments: Explain the functionality of the code // Easier for other people to understand // Easier to maintain / debug / modify  Indentation: Easier to identify <u>structure</u> / <u>blocks</u> // identify <u>blocks</u> of code	2																				
2(b)	<table border="1" data-bbox="363 510 1265 1160"> <thead> <tr> <th data-bbox="363 510 1002 566">Feature</th> <th data-bbox="1002 510 1265 566">Answer</th> </tr> </thead> <tbody> <tr> <td data-bbox="363 566 1002 651">A line number containing an example of an integer assignment statement</td> <td data-bbox="1002 566 1265 651">8, 9, 10,12, 17, 34, 45</td> </tr> <tr> <td data-bbox="363 651 1002 736">A line number containing the start of a selection structure</td> <td data-bbox="1002 651 1265 736">14, 19, 23</td> </tr> <tr> <td data-bbox="363 736 1002 822">A line number containing the end of a selection structure</td> <td data-bbox="1002 736 1265 822">28, 29, 30</td> </tr> <tr> <td data-bbox="363 822 1002 875">The upper bound of the <code>Mark</code> array</td> <td data-bbox="1002 822 1265 875">100</td> </tr> <tr> <td data-bbox="363 875 1002 929">The number of dimensions of the <code>Mark</code> array</td> <td data-bbox="1002 875 1265 929">1</td> </tr> <tr> <td data-bbox="363 929 1002 983">The name for the type of loop structure used</td> <td data-bbox="1002 929 1265 983">'post condition'</td> </tr> <tr> <td data-bbox="363 983 1002 1068">A line number containing an unnecessary assignment statement</td> <td data-bbox="1002 983 1265 1068">10</td> </tr> <tr> <td data-bbox="363 1068 1002 1117">The number of times that <code>OUTPUT</code> is called</td> <td data-bbox="1002 1068 1265 1117">100</td> </tr> <tr> <td data-bbox="363 1117 1002 1160">The number of local variables</td> <td data-bbox="1002 1117 1265 1160">4</td> </tr> </tbody> </table>	Feature	Answer	A line number containing an example of an integer assignment statement	8, 9, 10,12, 17, 34, 45	A line number containing the start of a selection structure	14, 19, 23	A line number containing the end of a selection structure	28, 29, 30	The upper bound of the <code>Mark</code> array	100	The number of dimensions of the <code>Mark</code> array	1	The name for the type of loop structure used	'post condition'	A line number containing an unnecessary assignment statement	10	The number of times that <code>OUTPUT</code> is called	100	The number of local variables	4	9
Feature	Answer																					
A line number containing an example of an integer assignment statement	8, 9, 10,12, 17, 34, 45																					
A line number containing the start of a selection structure	14, 19, 23																					
A line number containing the end of a selection structure	28, 29, 30																					
The upper bound of the <code>Mark</code> array	100																					
The number of dimensions of the <code>Mark</code> array	1																					
The name for the type of loop structure used	'post condition'																					
A line number containing an unnecessary assignment statement	10																					
The number of times that <code>OUTPUT</code> is called	100																					
The number of local variables	4																					
2(c)(i)	Either: <ul style="list-style-type: none"> <li>• Mistake: function header specifies return of an <code>INTEGER</code> but line 37 returns a <code>STRING</code> // pseudocode returns <code>Grade</code> but should have returned <code>DGradeCount</code></li> <li>• Correction: <code>RETURN DGradeCount</code> (as per code pseudocode comment)</li> </ul> Or: <ul style="list-style-type: none"> <li>• Mistake: Statement on line 32 uses <code>'&amp;'</code> operator which concatenates <code>STRINGS</code>, but variable <code>n</code> is an <code>INTEGER</code></li> <li>• Correction: Convert <code>n</code> to a <code>STRING</code> before concatenating</li> </ul>	2																				

Question	Answer	Marks
2(c)(ii)	<pre> CASE OF ThisMark     &gt; 74: Grade ← "Distinction"           DGradeCount ← DGradeCount + 1     60 TO 74: Grade ← "Merit"     40 TO 59: Grade ← "Pass"     OTHERWISE Grade ← "Fail" ENDCASE  One mark for each of:  1 CASE OF ThisMark ... ENDCASE 2 Three grade ranges with corresponding assignment of Grade 3 DGradeCount increment within CASE clause 4 OTHERWISE / fourth grade range with correct assignment of Grade                     </pre>	<b>4</b>

Question	Answer	Marks
3(a)	<p>Parameters</p> <p>Accept arguments</p>	<b>1</b>
3(b)	<div style="text-align: center;"> </div> <p>Mark as follows:</p> <ul style="list-style-type: none"> <li>• One mark for all four modules</li> <li>• One mark for each set of interface parameters</li> </ul>	<b>4</b>

Question	Answer	Marks
4	 <pre> graph TD     Start([START]) --&gt; InitIndex[Index ← 1]     InitIndex --&gt; InitCount[Count ← 0]     InitCount --&gt; LoopStart(( ))     LoopStart --&gt; Decision1{Is PTemp[Index] &lt; MinTemp? OR PTemp[Index] &gt; MaxTemp}     Decision1 -- YES --&gt; CountInc[Count ← Count + 1]     CountInc --&gt; IndexInc[Index ← Index + 1]     IndexInc --&gt; LoopStart     Decision1 -- NO --&gt; Decision2{Is Index = 1007}     Decision2 -- YES --&gt; Decision3{Is Count &lt;= 20}     Decision3 -- YES --&gt; RetValTrue[RetVal ← TRUE]     Decision3 -- NO --&gt; RetValFalse[RetVal ← FALSE]     Decision2 -- NO --&gt; IndexInc     RetValTrue --&gt; Return[RETURN RetVal]     RetValFalse --&gt; Return     Return --&gt; End([END])     </pre> <p>This is one possible solution – selection structure may differ</p> <p>One mark for:</p> <ol style="list-style-type: none"> <li>1 START <b>and</b> END // STOP</li> <li>2 Initialisation of an Index variable <b>and</b> initialisation of a Count variable</li> <li>3 Decision box / boxes to check temperature within acceptable range</li> <li>4 Correct increment of Count variable</li> <li>5 Decision box comparing Index to 100</li> <li>6 Correct increment of Index</li> <li>7 Decision box comparing Count &gt; 20</li> <li>8 Assigning both TRUE <b>and</b> FALSE</li> <li>9 Returning the Boolean value</li> </ol> <p>For solutions where Boolean variable not used:</p> <ol style="list-style-type: none"> <li>8 Return TRUE</li> <li>9 Return FALSE</li> </ol>	9

<b>Question</b>	<b>Answer</b>	<b>Marks</b>
5(a)(i)	1	<b>1</b>
5(a)(ii)	Information is saved after the program ends // after the computer is switched off	<b>1</b>
5(b)	Two from these examples: <ul style="list-style-type: none"><li>• Indentation</li><li>• Colour-coding of keywords /comments</li><li>• Expansion / collapsing of complex data structures</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> FUNCTION GetAverageScore (MembershipNumber : STRING)                                 RETURNS INTEGER      DECLARE FileData, FileMembershipNumber : STRING     DECLARE NumberOfScores, TotalScore, AverageScore :                                 INTEGER      OPENFILE "ScoreDetails.txt" FOR READ     NumberOfScores ← 0     TotalScore ← 0      WHILE NOT EOF("ScoreDetails.txt")         READFILE ("ScoreDetails.txt", FileData)         FileMembershipNumber ← LEFT(FileData, 4)         IF FileMembershipNumber = MembershipNumber             THEN                 NumberOfScores ← NumberOfScores + 1                 TotalScore ← TotalScore +                                 INT(RIGHT(FileData, 2))             ENDIF         ENDWHILE      AverageScore ← INT(TotalScore / NumberOfScores)      CLOSEFILE ("ScoreDetails.txt")      RETURN (AverageScore)  ENDFUNCTION </pre> <p>1 mark for each of the following:</p> <ol style="list-style-type: none"> <li>1 Function heading <b>and</b> ending including Input <b>and</b> return parameter</li> <li>2 Declare variables to store <code>NumberOfScores</code> <b>and</b> <code>TotalScore</code> as INTEGERS (commented in Python) (variable names may be different)</li> <li>3 Initialisation of <code>NumberOfScores</code> <b>and</b> <code>TotalScore</code> to 0</li> <li>4 Open file in READ mode</li> <li>5 Loop until EOF ( )</li> <li>6 Read a line from the file <b>in a loop</b></li> <li>7 Use of substring function to extract at least one data item</li> <li>8 Compare the membership number</li> <li>9 Convert score to an integer</li> <li>10 Increment <code>NumberOfScores</code> <b>and</b> sum <code>TotalScore</code></li> <li>11 Calculate the average outside the loop</li> <li>12 Close the file</li> <li>13 Return the parameter</li> </ol>	Max 10



Question	Answer	Marks
6(a)	Subscript / index	1
6(b)	<pre> FUNCTION Clip(MaxVal : INTEGER) RETURNS BOOLEAN   DECLARE i : INTEGER   DECLARE j : INTEGER   DECLARE ClipFlag : BOOLEAN    ClipFlag ← FALSE    FOR i ← 1 TO 8     FOR j ← 1 TO 8       IF Picture[i, j] &gt; MaxVal         THEN           Picture[i, j] ← MaxVal           ClipFlag ← TRUE         ENDFOR       ENDFOR     ENDFOR    RETURN ClipFlag  ENDFUNCTION </pre> <p>1 mark for each of the following:</p> <ol style="list-style-type: none"> <li>1 Correct Function heading (must have <code>MaxVal</code> and return a <code>BOOLEAN</code>) <b>and</b> ending</li> <li>2 Declare <b>and</b> initialise local variable for return <code>BOOLEAN</code> to <code>FALSE</code> / other mechanism to record pixel being clipped</li> <li>3 Declare local variables for loop counters</li> <li>4 Nested loops with correct number of iterations</li> <li>5 Accessing correct element from <code>Picture</code> array</li> <li>6 Comparing element with <code>MaxVal</code></li> <li>7 Changing value of element if necessary</li> <li>8 Setting flag to <code>TRUE</code> / other mechanism if element is changed</li> <li>9 Returning <code>BOOLEAN</code> <b>after loop</b> (following conversion if other mechanism used)</li> </ol>	9

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 IsFactor (Num1: INTEGER, Num2: INTEGER)                                 RETURNS BOOLEAN     IF Num2 &lt;&gt; 0       THEN         IF MOD (Num1, Num2) = 0           THEN             RETURN TRUE           ENDIF         ENDIF       ENDIF     RETURN FALSE ENDFUNCTION </pre> <p>1 mark for each of the following:</p> <ol style="list-style-type: none"> <li>1 Correct Function heading (including parameters) and ending</li> <li>2 Check that Num2 is not zero</li> <li>3 Mechanism to ensure no call to MOD (or equivalent) if Num2 is zero</li> <li>4 Use of MOD function or alternative</li> <li>5 Check value of remainder</li> <li>6 Return Boolean value</li> </ol>	6

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

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

```

Function GetAverageScore(ByVal MembershipNumber As String) As Integer

    Dim FileData As String
    Dim FileMembershipNumber As String
    Dim NumberOfScores As Integer
    Dim TotalScore As Integer
    Dim AverageScore As Integer
    Dim ObjReader As IO.StreamReader

    ObjReader = New IO.StreamReader("ScoreDetails.txt")
    NumberOfScores = 0
    TotalScore = 0

    Do While ObjReader.Peek <> -1
        FileData = ObjReader.ReadLine()
        FileMembershipNumber = LEFT(FileData, 4)
        If FileMembershipNumber = MembershipNumber Then
            NumberOfScores = NumberOfScores + 1
            TotalScore = TotalScore + INT(RIGHT(FileData, 2))
        End If
    Loop

    AverageScore = INT(TotalScore / NumberOfScores)
    ObjReader.Close()
    Return (AverageScore)

End Function

```

**Q5(c): Pascal**

```

function GetAverageScore(MembershipNumber : string):integer;

var
    FileData, FileMembershipNumber: string;
    NumberOfScores, TotalScore, AverageScore : integer;
    ScoreFile : textFile;
begin
    NumberOfScores := 0;
    TotalScore := 0;
    assignFile(ScoreFile, 'ScoreDetails.txt');
    reset(ScoreFile);
    while not eof(ScoreFile) do
        begin
            readln(ScoreFile, FileData);
            FileMembershipNumber := copy(FileData, 1, 4);
            if FileMembershipNumber = MembershipNumber then
                begin
                    NumberOfScores := NumberOfScores + 1
                    TotalScore := TotalScore + StrToInt(RightStr(FileData, 2));
                end;
            end;
        end;
end;

```

```
AverageScore := StrToInt(TotalScore / NumberOfScores);  
GetAverageScore := AverageScore;  
CloseFile (ScoreFile);  
end;
```

**Q5(c): Python**

```
# FileData AS STRING  
# FileMembershipNumber AS STRING  
# NumberOfScores AS INTEGER  
# TotalScore AS INTEGER  
# AverageScore AS INTEGER  
  
def GetAverageScore(MembershipNumber):  
    FileHandle = open("ScoreDetails.txt", "r")  
    NumberOfScores = 0  
    TotalScore = 0  
    FileData = FileHandle.readline()  
    while len(FileData) > 0:  
        FileMembershipNumber = FileData[0:4]  
        if FileMembershipNumber == MembershipNumber:  
            NumberOfScores = NumberOfScores + 1  
            TotalScore = TotalScore + int(FileData[-2])  
            FileData = FileHandle.readline()  
    AverageScore = int(TotalScore / NumberOfScores)  
    Return (AverageScore)  
    FileHandle.close()
```

**Q7: Visual Basic**

```
Function IsFactor(Num1 As Integer, Num2 As Integer) As Boolean
    If Num2 <> 0 Then
        If Num1 Mod Num2 = 0 Then
            Return True
        End If
    End if
    Return False
End Function
```

**Q7: Pascal**

```
function IsFactor(Num1,Num2 : integer) : boolean;

begin
    if Num2 <> 0 then
        begin
            if Num1 MOD Num2 = 0 then
                Return True;
            end;
            Return False;
        end;
    end;
```

**Q7: Python**

```
def IsFactor (Num1, Num2):
    if Num2 != 0:
        if Num1 % Num2 == 0:
            Return True
    Return False
```