

---

**COMPUTER SCIENCE**

**9608/23**

Paper 2 Written Paper

**October/November 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 October/November 2018 series for most Cambridge IGCSE™, Cambridge International A and AS Level components and some Cambridge O Level components.

---

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)(i)	<table border="1"> <thead> <tr> <th data-bbox="320 248 796 313">Statement</th> <th data-bbox="801 248 991 313">Assignment</th> <th data-bbox="995 248 1155 313">Selection</th> <th data-bbox="1160 248 1310 313">Iteration</th> </tr> </thead> <tbody> <tr> <td data-bbox="320 320 796 385">CASE OF TempSensor1</td> <td data-bbox="801 320 991 385"></td> <td data-bbox="995 320 1155 385">✓</td> <td data-bbox="1160 320 1310 385"></td> </tr> <tr> <td data-bbox="320 392 796 456">ELSE</td> <td data-bbox="801 392 991 456"></td> <td data-bbox="995 392 1155 456">✓</td> <td data-bbox="1160 392 1310 456"></td> </tr> <tr> <td data-bbox="320 463 796 528">REPEAT</td> <td data-bbox="801 463 991 528"></td> <td data-bbox="995 463 1155 528"></td> <td data-bbox="1160 463 1310 528">✓</td> </tr> <tr> <td data-bbox="320 535 796 600">ENDFOR</td> <td data-bbox="801 535 991 600"></td> <td data-bbox="995 535 1155 600"></td> <td data-bbox="1160 535 1310 600">✓</td> </tr> <tr> <td data-bbox="320 607 796 672">DayNumber ← DayNumber + 1</td> <td data-bbox="801 607 991 672">✓</td> <td data-bbox="995 607 1155 672"></td> <td data-bbox="1160 607 1310 672"></td> </tr> <tr> <td data-bbox="320 678 796 743">Error ← TRUE</td> <td data-bbox="801 678 991 743">✓</td> <td data-bbox="995 678 1155 743"></td> <td data-bbox="1160 678 1310 743"></td> </tr> </tbody> </table> <p data-bbox="316 741 555 775">One mark per row</p>				Statement	Assignment	Selection	Iteration	CASE OF TempSensor1		✓		ELSE		✓		REPEAT			✓	ENDFOR			✓	DayNumber ← DayNumber + 1	✓			Error ← TRUE	✓			6
Statement	Assignment	Selection	Iteration																														
CASE OF TempSensor1		✓																															
ELSE		✓																															
REPEAT			✓																														
ENDFOR			✓																														
DayNumber ← DayNumber + 1	✓																																
Error ← TRUE	✓																																
1(b)(i)	<table border="1"> <thead> <tr> <th data-bbox="347 804 1023 869">Statement</th> <th data-bbox="1027 804 1281 869">Data type</th> </tr> </thead> <tbody> <tr> <td data-bbox="347 875 1023 940">Revision ← 500</td> <td data-bbox="1027 875 1281 940">INTEGER</td> </tr> <tr> <td data-bbox="347 947 1023 1012">FuelType ← 'P'</td> <td data-bbox="1027 947 1281 1012">CHAR</td> </tr> <tr> <td data-bbox="347 1019 1023 1084">MinValue ← -6.3</td> <td data-bbox="1027 1019 1281 1084">REAL</td> </tr> <tr> <td data-bbox="347 1090 1023 1155">ServiceDue ← FALSE</td> <td data-bbox="1027 1090 1281 1155">BOOLEAN</td> </tr> <tr> <td data-bbox="347 1162 1023 1227">ModelRef ← "W212DEC15"</td> <td data-bbox="1027 1162 1281 1227">STRING</td> </tr> </tbody> </table> <p data-bbox="316 1256 555 1290">One mark per row</p>		Statement	Data type	Revision ← 500	INTEGER	FuelType ← 'P'	CHAR	MinValue ← -6.3	REAL	ServiceDue ← FALSE	BOOLEAN	ModelRef ← "W212DEC15"	STRING		5																	
Statement	Data type																																
Revision ← 500	INTEGER																																
FuelType ← 'P'	CHAR																																
MinValue ← -6.3	REAL																																
ServiceDue ← FALSE	BOOLEAN																																
ModelRef ← "W212DEC15"	STRING																																
1(b)(ii)	<table border="1"> <thead> <tr> <th data-bbox="347 1319 1023 1384">Expression</th> <th data-bbox="1027 1319 1281 1384">Evaluates to</th> </tr> </thead> <tbody> <tr> <td data-bbox="347 1391 1023 1456">"Month: " &amp; MID(ModelRef, 5, 3)</td> <td data-bbox="1027 1391 1281 1456">"Month: DEC"</td> </tr> <tr> <td data-bbox="347 1462 1023 1527">INT(MinValue * 2)</td> <td data-bbox="1027 1462 1281 1527">-12</td> </tr> <tr> <td data-bbox="347 1534 1023 1599">ASC(Revision)</td> <td data-bbox="1027 1534 1281 1599">ERROR</td> </tr> <tr> <td data-bbox="347 1606 1023 1671">Revision &gt; 500</td> <td data-bbox="1027 1606 1281 1671">FALSE</td> </tr> <tr> <td data-bbox="347 1677 1023 1742">ServiceDue = TRUE OR FuelType = 'P'</td> <td data-bbox="1027 1677 1281 1742">TRUE</td> </tr> </tbody> </table> <p data-bbox="316 1749 555 1783">One mark per row</p>		Expression	Evaluates to	"Month: " & MID(ModelRef, 5, 3)	"Month: DEC"	INT(MinValue * 2)	-12	ASC(Revision)	ERROR	Revision > 500	FALSE	ServiceDue = TRUE OR FuelType = 'P'	TRUE		5																	
Expression	Evaluates to																																
"Month: " & MID(ModelRef, 5, 3)	"Month: DEC"																																
INT(MinValue * 2)	-12																																
ASC(Revision)	ERROR																																
Revision > 500	FALSE																																
ServiceDue = TRUE OR FuelType = 'P'	TRUE																																

Question	Answer	Marks
2(a)(i)	<pre> FUNCTION CalcBonus(CardNum: STRING) RETURNS INTEGER    DECLARE Points : INTEGER   DECLARE Bonus : INTEGER   DECLARE Spend : REAL    Points ← GetPoints(CardNum)   Spend ← GetSpend(CardNum)    IF Points &gt; 2000     THEN       Bonus ← 100     ELSE       IF Spend &gt; 1000         THEN           Bonus ← 50         ELSE           Bonus ← 10       ENDIF     ENDIF    RETURN Bonus  ENDFUNCTION </pre> <p>1 mark for each of the following up to max 5 marks:</p> <ol style="list-style-type: none"> <li>1 Function heading (inc parameters) <b>and</b> ending</li> <li>2 Declaring local variables and two function calls as above</li> <li>3 IF...THEN...ELSE...ENDIF with Points &gt; 2000</li> <li>4 (Nested) IF...THEN...ELSE with Spend &gt; 1000</li> <li>5 ... assignment of Bonus to 10, 50 100</li> <li>6 Return parameter</li> </ol>	<b>5</b>

Question	Answer	Marks
2(a)(ii)	<p>The pseudocode shown here is only an example. The use of an explicit flag and IF structure are not essential provided the functionality is provided.</p> <pre> FUNCTION GetCardNumber() RETURNS STRING      DECLARE Valid : BOOLEAN     DECLARE CardNum : STRING      Valid ← FALSE      REPEAT         OUTPUT "Enter card number"         INPUT CardNum         IF LENGTH(CardNum) = 16 AND IS_NUM(CardNum) = TRUE             THEN                 Valid ← TRUE             ENDIF         UNTIL Valid      RETURN CardNum  ENDFUNCTION </pre> <p>1 mark for each of the following:</p> <ol style="list-style-type: none"> <li>1 Declaring local variable to store user input</li> <li>2 Conditional Loop</li> <li>3 Prompt and input of CardNum</li> <li>4 Length check</li> <li>5 Checking IS_NUM(CardNum) is TRUE</li> <li>6 Return a value</li> </ol>	6
2(b)(i)	<p>Name: Logic (error) Description: Where the program does not behave as expected / Does not give expected result / An error in the logic of the algorithm</p> <p><b>OR</b></p> <p>Name: Run-time // execution (error) Description: The program performs an illegal operation</p> <p>One mark for name + one mark for corresponding description</p>	2
2(b)(ii)	<p>Values: any Spend value and Points &gt; 2000 Justification: Bonus should be 100</p> <p>Values: Spend &gt; 1000 and Points ≤ 2000 Justification: Bonus should be 50</p> <p>Values: Spend ≤ 1000 and Points ≤ 2000 Justification: Bonus should be 10</p> <p>2 marks for values 2 marks for relevant and appropriate reasons</p>	4

Question	Answer	Marks
2(c)	<p>Name: Corrective Reason: Amend the algorithm to 'eliminate errors'</p> <p>Name: Adaptive Reason: In response to specification change arising from changes to business rules or environment (regulatory)</p> <p>Name: Perfective Reason: To make improvements to the program</p> <p>One mark for each name plus one mark for corresponding reason up to max 4 marks</p>	4

Question	Answer	Marks
3(a)	<p>Name: count controlled / FOR ... NEXT loop Justification: Known / fixed number of iterations // all elements of the array need to be checked</p> <p>1 mark for name 1 mark for justification</p>	2
3(b)	<p>Steps:</p> <ul style="list-style-type: none"> <li>• Declare (and initialise values to first array element) for min and max as integers</li> <li>• A loop / iteration / repetition to check every element</li> <li>• Compare each array element with max variable and min variable</li> <li>• Update max variable if bigger and min variable if smaller</li> </ul> <p>1 mark per bullet point</p> <p>Alternative steps:</p> <ul style="list-style-type: none"> <li>• Apply a sort routine to the values in the array</li> <li>• Swapping consecutive elements (as necessary) // until no more swaps</li> <li>• Min will be the first / last element and max will be the last / first element</li> </ul> <p>1 mark per bullet point</p> <p>Max 3 marks</p>	3

Question	Answer		Marks			
4(a)(i)	The name of a global identifier	LastElement // ThisArray	5			
The name of a user-defined procedure	Insert					
The scope of ArrayIndex	Local					
The number of dimensions of ThisArray	2					
The scope of NewData	Local					
4(a)(ii)	<p>Example mark points:</p> <ul style="list-style-type: none"> <li>• Conditional loop through array <code>ThisArray</code> one element at a time until found</li> <li>• Compare the element from row / column 1 of the array with <code>NewData</code></li> <li>• If the current element is greater than <code>NewData</code> set <code>Found</code> to <code>TRUE</code> (to exit the loop)</li> <li>• If the current element is not greater than <code>NewData</code> increment <code>ArrayIndex</code></li> </ul>		4			

Question	Answer	Marks
4(b)	<p>'Pseudocode' solution included here for development and clarification of mark scheme.</p> <p>Programming language solutions appear at the end of this mark scheme.</p> <pre> FUNCTION Update(NewData: STRING) RETURNS INTEGER    DECLARE ArrayIndex : INTEGER   DECLARE Found : BOOLEAN   DECLARE Validate : BOOLEAN    ArrayIndex ← 1   Found ← FALSE    WHILE ArrayIndex &lt;= LastElement AND Found = FALSE     IF ThisArray[ArrayIndex, 1] &gt; NewData       THEN         Found ← TRUE       ELSE         ArrayIndex ← ArrayIndex + 1       ENDIF   ENDWHILE    IF Found = TRUE     THEN       Validate ← Insert(ArrayIndex, NewData)       IF Validate = FALSE         THEN           ArrayIndex ← -1         ENDIF       ELSE         ArrayIndex ← 0       ENDIF     RETURN ArrayIndex    ENDFUNCTION </pre> <p>1 mark for each of the following:</p> <ol style="list-style-type: none"> <li>1 Function heading and ending including parameters</li> <li>2 Local variable declarations and Initialisation of ArrayIndex and Found</li> <li>3 WHILE loop</li> <li>4 First IF-THEN-ELSE-ENDIF clause</li> <li>5 Second IF clause including function call to Insert()</li> <li>6 Check Return value</li> <li>7 Set / return -1 IF (Validate) FALSE</li> <li>8 Return parameter value (-1 or 0 // ArrayIndex)</li> </ol>	8



Question	Answer	Marks
4(c)	Description to include: <ul style="list-style-type: none"> <li>• mechanism involves using <u>parameters</u> .. to pass values from one procedure to another</li> <li>• parameters may be 'by reference' or 'by value'</li> </ul>	<b>2</b>
4(d)(i)	Pseudocode solution included here for development and clarification of mark scheme.  Programming language solutions appear at the end of this mark scheme.  <pre>           DECLARE i : INTEGER           FOR i ← 1 to 200             IF CharArray[i] &gt;= '0' AND CharArray[i] &lt;= '9'               THEN                 CharArray[i] ← '*'             ENDIF           ENDFOR           </pre> 1 mark for each of the following: <ul style="list-style-type: none"> <li>• looping through 200 elements</li> <li>• selection statement</li> <li>• assignment of '*'</li> </ul>	<b>3</b>
4(d)(ii)	CONSTANT LastElement = 200	<b>1</b>

Question	Answer	Marks
5	<pre> FUNCTION ReadFileLine (FileName: STRING,                         FileLine: INTEGER) RETURNS STRING    DECLARE FileData : STRING   DECLARE LineNumber : INTEGER    OPENFILE FileName FOR READ    LineNumber ← 0 // no line read yet    WHILE (NOT EOF(FileName)) AND FileLine &lt;&gt; LineNumber     READFILE FileName, FileData     LineNumber ← LineNumber + 1   ENDWHILE    IF FileLine &lt;&gt; LineNumber     THEN       FileData ← "*****"     ENDIF    CLOSEFILE FileName    RETURN FileData  ENDFUNCTION </pre> <p>1 mark for each of the following:</p> <ol style="list-style-type: none"> <li>1 Function heading including parameters.</li> <li>2 Declare local variables FileData and LineNumber</li> <li>3 Open FileName in READ mode</li> <li>4 WHILE loop</li> <li>5 Call to READFILE () (in a loop)</li> <li>6 Incrementing LineNumber (in a loop)</li> <li>7 IF FileLine &lt;&gt; LineNumber (after a loop)</li> <li>8 ...Set FileData to "*****"</li> <li>9 Close FileName</li> <li>10 Return FileData</li> </ol>	10

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

**Program Code Example Solutions****Q4(b): Visual Basic**

```
FUNCTION Update (ByVal NewData AS STRING) AS INTEGER
```

```
    DIM ArrayIndex AS INTEGER
    DIM Found AS BOOLEAN
    DIM Validate AS BOOLEAN
```

```
    ArrayIndex = 1
    Found = FALSE
```

```
    WHILE ArrayIndex <= LastElement AND Found = FALSE
        IF ThisArray[ArrayIndex, 1] > NewData
            THEN
                Found = TRUE
            ELSE
                ArrayIndex = ArrayIndex + 1
            ENDIF
    ENDWHILE
```

```
**IF Found = TRUE
    THEN
        Validate = Insert(ArrayIndex, NewData)
        IF Validate = FALSE
            THEN
                ArrayIndex = -1
            ENDIF
    ELSE
        ArrayIndex = 0
    ENDIF

    RETURN ArrayIndex // Update = ArrayIndex
```

```
ENDFUNCTION
```

**\*\* Alternative**

```
    IF Found = FALSE
        THEN
            ArrayIndex = 0
        ELSE
            Validate = Insert(ArrayIndex, NewData)
            IF Validate = FALSE
                THEN
                    ArrayIndex = -1
                ENDIF
        ENDIF

    RETURN ArrayIndex
```

**Q4(b): Pascal**

```
function Update(NewData: string): integer;
```

```

var ArrayIndex : integer;
var Found : boolean;
var Validate : boolean;

begin
  ArrayIndex := 1;
  Found := FALSE;

  while ArrayIndex <= LastElement AND Found = FALSE do
    begin
      if ThisArray[ArrayIndex, 1] > NewData then found := True
      else ArrayIndex := ArrayIndex + 1;

    end;

  if Found = TRUE then
    begin
      Validate := Insert(ArrayIndex, NewData);
      if Validate = FALSE then
        ArrayIndex := -1;
      end
    else
      begin
        ArrayIndex := 0;
      end;

    Update := ArrayIndex;
  end;

```

**Q4(b): Python**

```

def Update(NewData):

    # ArrayIndex AS INTEGER
    # Found AS BOOLEAN
    # Validate AS BOOLEAN

    ArrayIndex = 1
    Found = FALSE
    LastElement = 20

    while ArrayIndex <= LastElement AND Found == FALSE:
        if ThisArray[ArrayIndex][1] > NewData:
            Found = TRUE
        else:
            ArrayIndex = ArrayIndex + 1

        if Found == TRUE:
            Validate = Insert(ArrayIndex, NewData)
            if Validate == FALSE:
                ArrayIndex = -1
            else:
                ArrayIndex = 0

    return ArrayIndex

```

**Q4(d)(i): Visual Basic**

```
Dim i AS Integer

For i = 1 to 200
    If CharArray(i) >= '0' AND CharArray(i) <= '9' then
        CharArray(i) = '*'
    Endif
Next i
```

**Q4(d)(i): Pascal**

```
var i : integer;

for i := 1 to 200 do
    begin
        If CharArray(i) >= '0' AND CharArray(i) <= '9' then
            CharArray(i) := '*';
        end;
    end;
```

**Q4(d)(i): Python**

```
#i as string
for i in CharArray:
    if CharArray.isdigit():
        i = '*'
```

**\*\* Alternative**

```
# i as integer

for i in range(200):
    if CharArray[i] >= '0' and CharArray[i] <= '9':
        CharArray[i] = '*'
```