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.
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 1(a)(i)

<table>
<thead>
<tr>
<th>Statement</th>
<th>Assignment</th>
<th>Selection</th>
<th>Iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>CASE OF TempSensor1</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>ELSE</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>REPEAT</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>ENDFOR</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>DayNumber ← DayNumber + 1</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Error ← TRUE</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>

One mark per row

### Question 1(b)(i)

<table>
<thead>
<tr>
<th>Statement</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Revision ← 500</td>
<td>INTEGER</td>
</tr>
<tr>
<td>FuelType ← 'P'</td>
<td>CHAR</td>
</tr>
<tr>
<td>MinValue ← -6.3</td>
<td>REAL</td>
</tr>
<tr>
<td>ServiceDue ← FALSE</td>
<td>BOOLEAN</td>
</tr>
<tr>
<td>ModelRef ← &quot;W212DEC15&quot;</td>
<td>STRING</td>
</tr>
</tbody>
</table>

One mark per row

### Question 1(b)(ii)

<table>
<thead>
<tr>
<th>Expression</th>
<th>Evaluates to</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Month: &quot; &amp; MID(ModelRef, 5, 3)</td>
<td>&quot;Month: DEC&quot;</td>
</tr>
<tr>
<td>INT(MinValue * 2)</td>
<td>-12</td>
</tr>
<tr>
<td>ASC(Revision)</td>
<td>ERROR</td>
</tr>
<tr>
<td>Revision &gt; 500</td>
<td>FALSE</td>
</tr>
<tr>
<td>ServiceDue = TRUE OR FuelType = 'P'</td>
<td>TRUE</td>
</tr>
</tbody>
</table>

One mark per row
<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
<th>Marks</th>
</tr>
</thead>
<tbody>
<tr>
<td>2(a)(i)</td>
<td>FUNCTION CalcBonus(CardNum: STRING) RETURNS INTEGER</td>
<td>5</td>
</tr>
</tbody>
</table>

```plaintext
DECLARE Points : INTEGER
DECLARE Bonus : INTEGER
DECLARE Spend : REAL

Points ← GetPoints(CardNum)
Spend ← GetSpend(CardNum)

IF Points > 2000
    THEN
        Bonus ← 100
    ELSE
        IF Spend > 1000
            THEN
                Bonus ← 50
            ELSE
                Bonus ← 10
            ENDIF
    ENDIF
RETURN Bonus
ENDFUNCTION
```

1 mark for each of the following up to max 5 marks:

1 Function heading (inc parameters) and ending
2 Declaring local variables and two function calls as above
3 IF...THEN...ELSE...ENDIF with Points > 2000
4 (Nested) IF...THEN...ELSE with Spend > 1000
5 ... assignment of Bonus to 10, 50, 100
6 Return parameter
<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
<th>Marks</th>
</tr>
</thead>
<tbody>
<tr>
<td>2(a)(ii)</td>
<td>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.</td>
<td>6</td>
</tr>
</tbody>
</table>

```plaintext
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
```

1 mark for each of the following:
1. Declaring local variable to store user input
2. Conditional Loop
3. Prompt and input of CardNum
4. Length check
5. Checking IS_NUM(CardNum) is TRUE
6. Return a value

| 2(b)(i) | 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 OR Name: Run-time // execution (error) Description: The program performs an illegal operation | 2 |

| 2(b)(ii) | Values: any Spend value and Points > 2000 Justification: Bonus should be 100 Values: Spend > 1000 and Points <= 2000 Justification: Bonus should be 50 Values: Spend <= 1000 and Points <= 2000 Justification: Bonus should be 10 | 4 |

2 marks for values
2 marks for relevant and appropriate reasons
<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
<th>Marks</th>
</tr>
</thead>
</table>
| 2(c)     | Name: Corrective  
          Reason: Amend the algorithm to ‘eliminate errors’  
          Name: Adaptive  
          Reason: In response to specification change arising from changes to  
          business rules or environment (regulatory)  
          Name: Perfective  
          Reason: To make improvements to the program  
          One mark for each name plus one mark for corresponding reason up to max 4 marks | 4 |
| 3(a)     | Name: count controlled / FOR ... NEXT loop  
          Justification: Known / fixed number of iterations // all elements of the array need to be checked  
          1 mark for name  
          1 mark for justification | 2 |
| 3(b)     | Steps:  
          • Declare (and initialise values to first array element) for min and max as integers  
          • A loop / iteration / repetition to check every element  
          • Compare each array element with max variable and min variable  
          • Update max variable if bigger and min variable if smaller  
          1 mark per bullet point  
          Alternative steps:  
          • Apply a sort routine to the values in the array  
          • Swapping consecutive elements (as necessary) // until no more swaps  
          • Min will be the first / last element and max will be the last / first element  
          1 mark per bullet point  
          Max 3 marks | 3 |
<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
<th>Marks</th>
</tr>
</thead>
<tbody>
<tr>
<td>4(a)(i)</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>The name of a global identifier</td>
<td>LastElement // ThisArray</td>
<td></td>
</tr>
<tr>
<td>The name of a user-defined procedure</td>
<td>Insert</td>
<td></td>
</tr>
<tr>
<td>The scope of ArrayIndex</td>
<td>Local</td>
<td></td>
</tr>
<tr>
<td>The number of dimensions of ThisArray</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>The scope of NewData</td>
<td>Local</td>
<td></td>
</tr>
<tr>
<td>4(a)(ii)</td>
<td>Example mark points:</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>- Conditional loop through array ThisArray one element at a time until found</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Compare the element from row / column 1 of the array with NewData</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- If the current element is greater than NewData set Found to TRUE (to exit the loop)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- If the current element is not greater than NewData increment ArrayIndex</td>
<td></td>
</tr>
</tbody>
</table>
4(b) ‘Pseudocode’ solution included here for development and clarification of mark scheme.

Programming language solutions appear at the end of this mark scheme.

FUNCTION Update(NewData: STRING) RETURNS INTEGER

DECLARE ArrayIndex : INTEGER
DECLARE Found : BOOLEAN
DECLARE Validate : BOOLEAN

ArrayIndex ← 1
Found ← FALSE

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

IF Found = TRUE
   Validate ← Insert(ArrayIndex, NewData)
   IF Validate = FALSE
      ArrayIndex ← -1
   ENDIF
ELSE
   ArrayIndex ← 0
ENDIF
RETURN ArrayIndex
ENDFUNCTION

1 mark for each of the following:

1 Function heading and ending including parameters
2 Local variable declarations and Initialisation of ArrayIndex and Found
3 WHILE loop
4 First IF-THEN-ELSE-ENDIF clause
5 Second IF clause including function call to Insert()
6 Check Return value
7 Set / return –1 IF (Validate) FALSE
8 Return parameter value (–1 or 0 // ArrayIndex)
<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
<th>Marks</th>
</tr>
</thead>
</table>
| 4(c) | Description to include:  
• mechanism involves using **parameters** .. to pass values from one procedure to another  
• parameters may be 'by reference' or 'by value' | 2 |
| 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.  
DECLARE i : INTEGER  
FOR i ← 1 to 200  
    IF CharArray[i] >= '0' AND CharArray[i]<= '9'  
    THEN  
        CharArray[i] ← '∗'  
    ENDIF  
ENDFOR | 3 |
<p>| 4(d)(ii) | CONSTANT LastElement = 200 | 1 |</p>
<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
</table>
| 5        | 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 <> LineNumber  
          | READFILE FileName, FileData  
          | LineNumber ← LineNumber + 1  
          | ENDWHILE  
          | IF FileLine <> LineNumber  
          |   THEN  
          |   FileData ← "****"  
          | ENDIF  
          | CLOSEFILE FileName  
          | RETURN FileData  
          | ENDFUNCTION  

1 mark for each of the following:

1. Function heading including parameters.
2. Declare local variables FileData and LineNumber.
3. Open FileName in READ mode.
4. WHILE loop.
5. Call to READFILE() (in a loop).
6. Incrementing LineNumber (in a loop).
7. IF FileLine <> LineNumber (after a loop).
8. ...Set FileData to "****".
9. Close FileName.

*** 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
            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;
        else
            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;

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] = '*'