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 2019 series for most Cambridge IGCSE™, Cambridge International A and AS Level and Cambridge Pre-U 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.

<table>
<thead>
<tr>
<th>GENERIC MARKING PRINCIPLE 1:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Marks must be awarded in line with:</td>
</tr>
<tr>
<td>• the specific content of the mark scheme or the generic level descriptors for the question</td>
</tr>
<tr>
<td>• the specific skills defined in the mark scheme or in the generic level descriptors for the question</td>
</tr>
<tr>
<td>• the standard of response required by a candidate as exemplified by the standardisation scripts.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GENERIC MARKING PRINCIPLE 2:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Marks awarded are always <strong>whole marks</strong> (not half marks, or other fractions).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GENERIC MARKING PRINCIPLE 3:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Marks must be awarded <strong>positively</strong>:</td>
</tr>
<tr>
<td>• 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</td>
</tr>
<tr>
<td>• marks are awarded when candidates clearly demonstrate what they know and can do</td>
</tr>
<tr>
<td>• marks are not deducted for errors</td>
</tr>
<tr>
<td>• marks are not deducted for omissions</td>
</tr>
<tr>
<td>• 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.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GENERIC MARKING PRINCIPLE 4:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rules must be applied consistently e.g. in situations where candidates have not followed instructions or in the application of generic level descriptors.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GENERIC MARKING PRINCIPLE 5:</th>
</tr>
</thead>
<tbody>
<tr>
<td>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).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GENERIC MARKING PRINCIPLE 6:</th>
</tr>
</thead>
<tbody>
<tr>
<td>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.</td>
</tr>
<tr>
<td>Question</td>
</tr>
<tr>
<td>----------</td>
</tr>
<tr>
<td>1(a)(i)</td>
</tr>
<tr>
<td>1(a)(ii)</td>
</tr>
<tr>
<td>1(b)(i)</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>STRING_TO_NUM(RIGHT(ID, 3))</td>
</tr>
<tr>
<td>INT(Height * Children)</td>
</tr>
<tr>
<td>IsMarried AND Married &lt; 31/12/1999</td>
</tr>
<tr>
<td>LENGTH(ID &amp; NUM_TO_STRING(Height))</td>
</tr>
<tr>
<td>MID((ID, INT(Height) - Children, 2)&quot;23&quot;</td>
</tr>
<tr>
<td>No quotes for row 1&lt;br&gt;Quotes (single or double) for row 5</td>
</tr>
<tr>
<td>1(b)(ii)</td>
</tr>
<tr>
<td>Married</td>
</tr>
<tr>
<td>ID</td>
</tr>
<tr>
<td>MiddleInitial</td>
</tr>
<tr>
<td>Height</td>
</tr>
<tr>
<td>IsMarried</td>
</tr>
<tr>
<td>One mark per data type</td>
</tr>
<tr>
<td>Question</td>
</tr>
<tr>
<td>----------</td>
</tr>
<tr>
<td>2(a)(i)</td>
</tr>
<tr>
<td>2(a)(ii)</td>
</tr>
<tr>
<td>2(a)(iii)</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>2(b)</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
**Question** | **Answer** | **Marks**
--- | --- | ---
2(c) | **Control structure:** A (pre-) conditional loop  
**Function of code:**  
- Check if Result is less than 20 and if true, calls ResetSensor with parameter value 3...  
- ... and assign the value returned by GetSensor with parameter value 3 to Result  
- Loop until Result >= 20  
OR  
**Control structure:** A selection // conditional statement  
**Function of code:**  
- Check if Result is less than 20 and if true, calls ResetSensor with parameter value 3...  
- ... and assign the value returned by GetSensor with parameter value 3 to Result

One mark for control structure, maximum two for function  
Function of code marks independent of answer to control structure

<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
<th>Marks</th>
</tr>
</thead>
</table>
| 3(a)(i) | PROCEDURE SubA (A : STRING, B : INTEGER, BYREF C : CHAR)  
One mark for each underlined part  
Ignore BYVAL for parameter A and/or parameter B  
Parameter order / names not important but must be correct data types | 3 |
| 3(a)(ii) | Function SubB (D : STRING, E : INTEGER) RETURNS BOOLEAN  
One mark for each underlined part  
Ignore BYVAL for parameter D and/or parameter E  
Parameter order / names not important but must be correct data types | 3 |
| 3(b) | - Selection  
- Iteration  
- Sequence  
One mark per bullet to max. 2 | 2 |
### Question 4(a)(i)

<table>
<thead>
<tr>
<th>Index</th>
<th>NextChar</th>
<th>Selected</th>
<th>NewValue</th>
<th>NewString</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td>&quot;0&quot;</td>
</tr>
<tr>
<td>1</td>
<td>'1'</td>
<td></td>
<td></td>
<td>&quot;01&quot;</td>
</tr>
<tr>
<td>2</td>
<td>'2'</td>
<td></td>
<td></td>
<td>&quot;012&quot;</td>
</tr>
<tr>
<td>3</td>
<td>'∇'</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>12</td>
<td></td>
<td>&quot;0&quot;</td>
</tr>
<tr>
<td>4</td>
<td>'3'</td>
<td></td>
<td></td>
<td>&quot;03&quot;</td>
</tr>
<tr>
<td>5</td>
<td>'4'</td>
<td></td>
<td></td>
<td>&quot;034&quot;</td>
</tr>
<tr>
<td>6</td>
<td>'∇'</td>
<td>34</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>34</td>
<td></td>
<td>&quot;0&quot;</td>
</tr>
<tr>
<td>7</td>
<td>'5'</td>
<td></td>
<td></td>
<td>&quot;05&quot;</td>
</tr>
<tr>
<td>8</td>
<td>'∇'</td>
<td>5</td>
<td></td>
<td>&quot;0&quot;</td>
</tr>
<tr>
<td>9</td>
<td>'∇'</td>
<td>0</td>
<td></td>
<td>&quot;0&quot;</td>
</tr>
<tr>
<td>10</td>
<td>'3'</td>
<td></td>
<td></td>
<td>&quot;03&quot;</td>
</tr>
<tr>
<td>11</td>
<td>'9'</td>
<td></td>
<td></td>
<td>&quot;039&quot;</td>
</tr>
</tbody>
</table>

One mark for each column.

If no mark for columns, award one mark for initialisation of **Selected** to 0 and **NewString** to '0' (single or double quotes).

### Question 4(a)(ii)

<table>
<thead>
<tr>
<th>Answer</th>
<th>Marks</th>
</tr>
</thead>
<tbody>
<tr>
<td>34</td>
<td>1</td>
</tr>
<tr>
<td>Question</td>
<td>Answer</td>
</tr>
<tr>
<td>----------</td>
<td>--------</td>
</tr>
</tbody>
</table>
| 4(b)(i)  | • The final value (in the string) is the largest value (39) and is not considered // the final comparison with variable Selected is not made  
• The loop terminates at the end of the string (the character 9) // there wasn't a final space / non-numeric digit  
One mark per bullet. | 2 |
| 4(b)(ii) | • Check the (final) value of NewString after the loop...  
...and see if it is greater than Selected (repeat the existing conditional clause)  
OR  
• Amend the algorithm to add a space character / non-numeric character to the end of the string...  
...before the FOR loop / at the start of the function  
One mark per bullet point  
Accept alternative workable solution | 2 |
### Question 5(a)

**Answer**

One mark for each of:

- Open the file
- Set a count to zero
- Loop until end of file // no more lines to read
- Increment the count each time a line is read in a loop

**Marks**

3

---

### Question 5(b)

**Answer**

PROCEDURE CountLines(FileName : STRING)

```
DECLARE NumLines : INTEGER
DECLARE Dummy : STRING

NumLines ← 0

OPENFILE FileName FOR READ

WHILE NOT EOF(FileName)
    READFILE FileName, Dummy
    NumLines ← NumLines + 1
ENDWHILE

CLOSEFILE FileName

OUTPUT "Number of lines in the file : ", NumLines
```

**Marks**

6

---

**Answer**

One mark for each of the following:

1. Procedure header and end, including parameter
2. Declaration and initialisation of a local INTEGER to count lines (e.g. NumLines)
3. OPEN file in read mode and CLOSE file
4. WHILE loop stopping when EOF(FileName)
5. Read a line from the file and increment NumLines in a loop
6. Output a message plus the NumLines outside a loop
<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
<th>Marks</th>
</tr>
</thead>
<tbody>
<tr>
<td>6(a)</td>
<td>‘Pseudocode’ solution included here for development and clarification of mark scheme. Programming language example solutions appear in the Appendix. FUNCTION GetInfo() RETURNS STRING DECLARE ID : STRING DECLARE PreferredName : STRING DECLARE Valid : BOOLEAN Valid ← FALSE WHILE Valid = FALSE OUTPUT &quot;Please Enter a valid ID&quot; INPUT ID IF LENGTH(ID) = 5 AND LEFT(ID, 1) &gt;= 'A' AND LEFT(ID, 1) &lt;= 'Z' AND ISNUM(RIGHT(ID, 4)) THEN Valid ← TRUE ENDIF ENDFUNCTION</td>
<td>8</td>
</tr>
</tbody>
</table>

One mark for each of the following:

1. Function header and end (where appropriate)
2. Local variables used are declared (commented in python)
3. Prompt and input for ID (until valid) and preferred name
4. Conditional loop repeating while ID is invalid
5. test length in a loop
6. test first character in a loop
7. test last four characters in a loop
8. Concatenate using correct separator character and return resulting string
6(b) ‘Pseudocode’ solution included here for development and clarification of mark scheme.

Programming language example solutions appear in the Appendix.

PROCEDURE TopLevel()
    DECLARE Response : CHAR
    DECLARE InputData : STRING
    DECLARE Success : BOOLEAN

    Response ← 'Y'
    WHILE Response = 'Y'
        InputData ← GetInfo()
        IF LEFT(InputData,1) < 'N'
            THEN
                Success ← WriteInfo(InputData, "File1.txt")
            ELSE
                Success ← WriteInfo(InputData, "File2.txt")
            ENDIF
        IF NOT Success
            THEN
                Response ← 'N'
            ELSE
                OUTPUT "Enter details for another student? (Y/N)"
                INPUT Response
            ENDIF
    ENDWHILE
ENDPROCEDURE

One mark for each of the following:
1. Procedure header and end
2. Conditional loop terminated with user input
3. call to GetInfo() in a loop
4. check first character of returned UserID value in a loop
5. call(s) to WriteInfo() in both cases …
6. … with two STRING parameters in a loop
7. exit procedure if WriteInfo() unsuccessful in a loop
8. if WriteInfo() successful, prompt and check input to repeat / exit in a loop

6(c) FUNCTION WriteInfo (FileData : STRING, Filename : STRING)
RETURNS BOOLEAN

One mark per underlined section

*** End of Mark Scheme – example program code solutions follow ***
Program Code Example Solutions

Q6 (a): Visual Basic

Function GetInfo() As String
    Dim ID As String = ""
    Dim PreferredName As String = ""
    Dim Valid As Boolean = False
    While Valid = False
        Console.Write("Please enter a valid ID : ")
        ID = Console.ReadLine()
        If Len(ID) = 5 And Left(ID, 1) >= "A" And Left(ID, 1) <= "Z" __
            And IsNumeric(Right(ID, 4)) Then
            Valid = True
        End If
    End While
    Console.Write("Please enter preferred name : ")
    PreferredName = Console.ReadLine()
    Return ID & "*" & PreferredName
End Function

Alternative:

Function GetInfo() As String
    Dim ID As String
    Dim PreferredName As String
    Dim Valid As Boolean
    Dim Number As String
    Dim Size As Integer
    Dim i As Integer
    Valid = False
    While Valid = False
        Console.WriteLine("Please Enter a valid ID")
        ID = Console.ReadLine()
        Size = Len(ID)
        If (Size = 5) And ((Left(ID, 1) >= "A") And (Left(ID, 1) <= "Z") __
            And IsNumeric(Right(ID, 4)) Then
            Valid = True
        End If
        For i = 2 To 5
            Number = Mid(ID, i, 1)
            If (Number < "0") Or (Number > "9") Then
                Valid = False
            End If
        Next
    End While
    Console.WriteLine("Please enter preferred name")
    PreferredName = Console.ReadLine()
    Return (ID & "*" & PreferredName)
End Function
Q6 (a): Pascal

```pascal
function GetInfo() : String;
var
    ID : String;
    PreferredName : String;
    Valid : Boolean;
    Value, Code : Integer;
begin
    Valid := false;
    while not Valid do
    begin
        Write('Please enter a valid ID : ');
        Readln(ID);
        if (Length(ID) = 5) and (ID[1] >= 'A') and (ID[1] <= 'Z') then
            Valid := true;
        Val(Copy(ID, 2, 4), Value, Code);
        if Code <> 0 then
            Valid := false;
    end;
    Write('Please enter preferred name : ');
    Readln(PreferredName);
    GetInfo := ID + '*' + PreferredName;
end;
```

Free Pascal

```pascal
function GetInfo() : String;
var
    ID : String;
    PreferredName : String;
    Valid : Boolean;
    Value, Code : Integer;
begin
    Valid := false;
    while not Valid do
    begin
        Write('Please enter a valid ID : ');
        Readln(ID);
        if (Length(ID) = 5) and (ID[1] >= 'A') and (ID[1] <= 'Z') ___
            and (IsNumber(SubStr(ID, 2, 4))) then
            Valid := true;
    end;
    Write('Please enter preferred name : ');
    Readln(PreferredName);
    result := ID + '*' + PreferredName;
end;
```
Q6 (a): Python

def GetInfo() :
    ID = ""   # string variable
    PreferredName = "" # string variable
    Valid = False # Boolean variable
    while not Valid :
        ID = input("Please enter a valid ID : ")
        if len(ID) == 5 and ID[0] >= "A" and ID[0] <= "Z" and ID[1:].isnumeric() :
            Valid = True
        PreferredName = input("Please enter preferred name : ")
    return ID + "*" + PreferredName
Q6 (b): Visual Basic

Sub TopLevel()
    Dim Response As String = "Y"
    Dim InputData As String = ""
    Dim Success As Boolean = True
    While Response = "Y"
        InputData = GetInfo()
        If Left(InputData, 1) < "N" Then
            Success = WriteInfo(InputData, "File1.txt")
        Else
            Success = WriteInfo(InputData, "file2.txt")
        End If
        If Not Success Then
            Response = "N"
        Else
            Console.Write("Enter details for another student? Y/N ")
            Response = Console.ReadLine()
        End If
    End While
End Sub

Q6 (b): Pascal

procedure TopLevel();
var
    Response: Char;
    InputData: String;
    Success: Boolean;
begin
    Response := 'Y';
    while Response = 'Y' do
        begin
            InputData := GetInfo();
            if InputData[1] < 'N' then
                Success := WriteInfo(InputData, 'File1.txt')
            else
                Success := WriteInfo(InputData, 'File2.txt');
            if not Success then
                Response := 'N'
            else
                begin
                    Write('Enter details for another student? (Y/N) ');
                    Readln(Response);
                end;
        end;
end;
Q6 (b): Python

def TopLevel() :
    Response = "Y"      # string/character variable
    InputData = ""      # string variable
    Success = True     # Boolean variable
    while Response == "Y" :
        InputData = GetInfo()
        if InputData[0] < "N" :
            Success = WriteInfo(InputData, "File1.txt")
        else :
            Success = WriteInfo(InputData, "File2.txt")
        if not Success :
            Response = "Y"
        else :
            Response = input("Enter details for another student? (Y/N) ")