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 will not enter into discussions about these mark schemes.

Cambridge is publishing the mark schemes for the May/June 2016 series for most Cambridge IGCSE®, Cambridge International A and AS Level components and some Cambridge O Level components.
## Question 1 (a) (i)

<table>
<thead>
<tr>
<th>Item</th>
<th>Statement</th>
<th>Selection</th>
<th>Iteration</th>
<th>Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MyScore = 65</td>
<td></td>
<td></td>
<td>✔</td>
</tr>
<tr>
<td>2</td>
<td>FOR IndexVal = 0 TO 99</td>
<td>✔</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>MyArray[3] = ID(MyString,3,2)</td>
<td>✔</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>IF MyScore &gt;= 70 THEN</td>
<td>✔</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>ENDWHILE</td>
<td>✔</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>ELSE Message = &quot;Error&quot;</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>

One mark per row
Additional ticks in any row cancels that row

## Question 1 (a) (ii)

<table>
<thead>
<tr>
<th>Item</th>
<th>Purpose of statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Assign 65 to MyScore</td>
</tr>
<tr>
<td>2</td>
<td>(Start of) loop with loop counter starting from zero &amp; going to 99 / repeating 100 times</td>
</tr>
<tr>
<td>3</td>
<td>Assign 2 chars from position 3/4 in MyString to MyArray element 3/4</td>
</tr>
<tr>
<td>4</td>
<td>Test if MyScore is greater than or equal to 70</td>
</tr>
<tr>
<td>5</td>
<td>Marks the end of WHILE / precondition loop //Return to top of loop to check condition</td>
</tr>
<tr>
<td>6</td>
<td>If a condition is FALSE, variable Message is assigned the value &quot;ERROR&quot;</td>
</tr>
</tbody>
</table>

Exact wording not important
Explanation must refer to variables or values used in code (except for row 5)

## Question 1 (a) (iii)

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;D&quot; &amp; RIGHT(MyString, 4)</td>
<td>&quot;Dance&quot;</td>
</tr>
<tr>
<td>LEFT(RIGHT(MyString, 7), 3)</td>
<td>&quot;ten&quot;</td>
</tr>
</tbody>
</table>

Must have correct case
Quotation marks optional
<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
<th>Marks</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 (a)</td>
<td></td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlarmState</td>
<td>BOOLEAN</td>
<td>Alarm is set to ON or OFF</td>
</tr>
<tr>
<td>SensorValue</td>
<td>INTEGER</td>
<td>Value / number from sensor / as input by user // used in calculation of Temperature</td>
</tr>
<tr>
<td>ThresholdValue</td>
<td>REAL / FLOAT / SINGLE / DOUBLE</td>
<td>Threshold value for comparison</td>
</tr>
<tr>
<td>Temperature</td>
<td>REAL / FLOAT / SINGLE / DOUBLE</td>
<td>Temperature value calculated from sensor value</td>
</tr>
</tbody>
</table>

One mark per row
Data types as shown
Descriptions given above are examples only
<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
<th>Marks</th>
</tr>
</thead>
<tbody>
<tr>
<td>(b)</td>
<td></td>
<td>Max 6</td>
</tr>
<tr>
<td></td>
<td>AlarmState ← FALSE 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>INPUT ThresholdValue</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>INPUT SensorValue</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Temperature ← SensorValue * 1.135 3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IF Temperature &gt; ThresholdValue THEN</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AlarmState ← TRUE 5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>OUTPUT &quot;Temperature Alarm&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ELSE 6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>OUTPUT &quot;Temperature OK&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AlarmState ← FALSE 7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ENDIF</td>
<td></td>
</tr>
</tbody>
</table>

Mark points as circled, descriptions as below:

1. Setting `AlarmState` to `FALSE` (Cond. check not essential but must be correct if present)
2. Inputting `SensorVal` and `ThresholdVal`
3. Correct value assigned to `Temperature` (must be * not x)
4. IF..THEN..ELSE..ENDIF structure with correct condition (or two separate IF clauses)
5. Correct THEN statements as shown
6. Correct ELSE statement as shown
7. Setting `AlarmState` to `FALSE` within ELSE clause only if mark point 1 not given
<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
<th>Marks</th>
</tr>
</thead>
</table>
| 3 (a) | FUNCTION EncryptString (LookUp : ARRAY, PlainText : STRING) RETURNS STRING  

DECLARE OldChar, NewChar : CHAR  
DECLARE OldCharValue : INTEGER  
DECLARE OutString: STRING  

// first initialise the return string  
OutString ← ""  
// initialise the return string  
// loop through PlainText to produce OutString  
FOR n ← 1 to LENGTH(PlainText) // from first to last character  
OldChar ← MID(PlainText, n, 1) // get next character  
OldCharValue ← ASC(OldChar) // find the ASCII value  
NewChar ← LookUp[OldCharValue] // look up substitute character  
OutString ← OutString & NewChar // concatenate to OutString  
ENDFOR  
RETURN OutString // EncryptString ← OutString  
ENDFUNCTION  

One mark for each part-statement (shown underlined and bold) |
| (b) (i) | VB: Dim Lookup(0 to 127 / 128) As CHAR  
Pascal: Var Lookup: Array[0..127 / 1..128] Of CHAR  
Python:  

Lookup = ["" for i in range(128)]  

OR  

Lookup = []  
For i in range(128):  
Lookup.append("")  

Mark as follows:  
VB / Pascal: one mark per part-statement as underlined and bold  
Python:  
One mark for Lookup and []  
One mark for range(128) | 2 |
(ii) 'Pseudocode' solution included here for development and clarification of mark scheme. Programming language solutions appear in the Appendix.

```plaintext
INPUT StartPos
INPUT NumToChange

FOR n = 0 to NumToChange - 1
    OUTPUT " Input new value for position "  
    INPUT NewChar
    Lookup[StartPos + n] ← NewChar
ENDFOR

OUTPUT (NumToChange & " entries changed")
```

ALTERNATIVE:

```plaintext
INPUT StartPos
INPUT NumToChange
n = 0

REPEAT
    OUTPUT " Input new value for position "  
    INPUT NewChar
    Lookup[StartPos + n] ← NewChar
    n ← n + 1
UNTIL n = NumToChange

OUTPUT (NumToChange & " entries changed")
```

Mark points as circled, descriptions as below:

1. Two INPUT statements
2. Working loop (using values of n from flowchart)
3. OUTPUT prompt (exact text not specified)
4. INPUT NewChar
5. Assignment of NewChar to correct array element
6. OUTPUT final message after loop (exact text not specified but must include NumToChange or loop counter if value correct at that point)

---

4 (a) - Program code is easier to implement / manage
   - Modules may be given to different people to develop // given to program specialists
   - Program code is easier to test / debug / maintain
   - Encourages the re-usability of program code

Max 2
<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
<th>Marks</th>
</tr>
</thead>
<tbody>
<tr>
<td>(b) (i)</td>
<td><img src="diagram.png" alt="Diagram" /></td>
<td>3</td>
</tr>
</tbody>
</table>

One mark per correct annotation as shown  
Arrows may be drawn clockwise or anticlockwise  
Diamond symbol may be filled or unfilled but must be in position shown

| (ii) | A (or B) – Card details / Card number / Card info  
B (or A) – Cost details / amount payable / product cost / total bill  
C – (Flag) indicator for successful payment // payment confirmation  
Data items for A and B are interchangeable | 3 |

| 5 (a) (i) | • So that the data / information is saved after the program is run / when the computer is switched off  
• So the data / information can be accessed next time the program is run  
• So the data information can be "permanently stored" | Max 1 |
(ii) Problem:
- When retrieving / searching for / editing (text relating to a particular CD)
- Can’t tell where the artist name stops and the title begins (or any similar explanation or example)

Solution 1:
- Use of a separator character// or by example
- Where the separator character does not occur in the original strings

Solution 2:
- Use a fixed number of characters for each data item
- Data items are padded with e.g. <Space> character where needed

Solution 3:
- Convert original data items to CamelCase
- …and add a Space separator

Mark as follows:
Two marks for description of problem
Two marks for description of solution
### Question (b)

'Pseudocode' solution included here for development and clarification of mark scheme. Programming language solutions appear in the Appendix.

```plaintext
PROCEDURE InputData()

DECLARE CDTitle : STRING
DECLARE CDArtist : STRING
DECLARE CDLocation : STRING
DECLARE FileData : STRING

OPENFILE "MyMusic" FOR WRITE

OUTPUT "Input CD Title"
INPUT CDTitle

WHILE CDTitle <> "##"
    OUTPUT "Input CD Artist"
    INPUT CDArtist
    OUTPUT "Input CD Location"
    INPUT CDLocation
    FileData = CDTitle & ':' & CDArtist & ':' & CDLocation
    WRITEFILE "MyMusic.txt", FileData
OUTPUT "Input CD Title"
INPUT CDTitle

ENDWHILE

CLOSEFILE("MyMusic.txt ")

ENDPROCEDURE
```

One mark for each of the following:

- Procedure heading and ending
- Declaration of CDTitle, CDArtist and CDLocation
- Open file for writing (Allow MyMusic or MyMusic.txt)
- Working conditional loop structure including test for rogue value (including initial input of CDTitle)
- Input of three data values (CDTitle, CDArtist and CDLocation) inside a loop
- String concatenation of three variables inside a loop
- Write three variables in single line to file inside a loop
- Close file
- Use of string separator

Solutions may repeatedly OPEN – WRITE – CLOSE within the loop. In this case the first OPEN could be in WRITE or APPEND mode with all others in APPEND.

**Marks**

Max 8
### Question 6 (a)

<table>
<thead>
<tr>
<th>n</th>
<th>f</th>
<th>x</th>
<th>y</th>
<th>MID(String1, x, 1)</th>
<th>MID(String2, y, 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td>'R'</td>
<td>'R'</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
<td>'E'</td>
<td>'A'</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td></td>
<td>'E'</td>
<td>'R'</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>1</td>
<td></td>
<td>'T'</td>
<td>'R'</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>1</td>
<td></td>
<td>'R'</td>
<td>'R'</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td></td>
<td></td>
<td>'A'</td>
<td>'A'</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td></td>
<td></td>
<td>'C'</td>
<td>'C'</td>
</tr>
</tbody>
</table>

One mark per correct column

- Minus one mark if anything else on first row
- Column 2 – ‘4’ must not precede ‘4’ in column 1 (as shown by arrow)
- Letters must all be in upper case
- Ignore quotation symbol

### (b) (i)
- to search for a string within another string / String2 within String1
- to return the position of the start of String2 within String1 // by example

First mark point: allow locate / find / calculate position of

### (ii)
- Value: 0 / zero
- **Meaning:** String2 not found in String1

### (iii) Option 1
- It is possible to “fall off” the end of String1 (or by example) while string match (for example, String1 = "Retrace", String2 = "Raced")
- MID (String1, x, 1)// description of 'subscript out of range'

### Option 2
- If either string is empty then
- subscript out of range error // description

### Option 3
- If String1 found within String 2
- An endless loop will occur
Appendix - Program Code Solutions

3  (b)(ii):  VB.NET

Console.Write("Enter start position: ")
StartPos = Console.ReadLine()
Console.Write("Enter number to change: ")
NumToChange = Console.ReadLine()
For n = 0 To NumToChange - 1
    Console.Write("Enter new value for position: ")
    NewChar = Console.ReadLine()
    Lookup(StartPos + n) = NewChar
Next
Console.WriteLine(NumToChange & " entries changed")

ALTERNATIVE:

Console.Write("Enter start position: ")
StartPos = Console.ReadLine()
Console.Write("Enter number to change: ")
NumToChange = Console.ReadLine()
n = 0
Do
    Console.Write("Enter new value for position: ")
    NewChar = Console.ReadLine()
    Lookup(StartPos + n) = NewChar
    n = n + 1
Loop Until n = NumToChange
Console.WriteLine(NumToChange & " entries changed")

3  (b)(ii):  Pascal

write('Enter start position: '); readln(StartPos);
write('Enter number to change: '); readln(NumToChange);
for n := 0 to NumToChange - 1 do begin
    write('Input new value for position: '); readln(NewChar);
    LookUp[Startpos + n] := NewChar;
end;
writeln(IntToStr(NumToChange) + ' entries changed');
ALTERNATIVE:

```plaintext
write('Enter start position: ');
readln(StartPos);
write('Enter number to change: ');
readln(NumToChange);
n := 0;
repeat
    write('Input new value for position: ');
    readln(NewChar);
    LookUp[Startpos + n] := NewChar;
    n := n + 1;
until (n = NumToChange);
writeln(IntToStr(NumToChange) + ' entries changed');
```

3 (b)(ii): Python

```python
StartPos = int(input("Enter start position: "))
NumToChange = int(input("Enter number to change: "))
for n in range(NumToChange):
    NewChar = input("Input new value for position: ")
    LookUp[StartPos + n - 1] = NewChar
print(str(NumToChange) + " entries changed")
```

ALTERNATIVE:

```python
StartPos = int(input("Enter start position: "))
NumToChange = int(input("Enter number to change: "))
n = 0
while n < NumToChange:
    NewChar = input("Input new value for position: ")
    LookUp[StartPos + n] = NewChar
    n = n + 1
print(str(NumToChange) + " entries changed")
```
5  (b): VB.NET

A StreamWriter() solution:

Sub InputData()
    Dim CDTitle, CDArtist, CDLocation As String
    Dim FileHandle As IO.StreamWriter
    FileHandle = New IO.StreamWriter("MyMusic.txt")
    Console.WriteLine("Input CD Title: ")
    CDTitle = Console.ReadLine()
    Do Until CDTitle = "##"
        Console.WriteLine("Input CD Artist: ")
        CDArtist = Console.ReadLine()
        Console.WriteLine("Input CD Location: ")
        CDLocation = Console.ReadLine()
        FileHandle.WriteLine(CDTitle & ":" & CDArtist & ":" & CDLocation)
        Console.WriteLine("Input CD Title: ")
        CDTitle = Console.ReadLine()
    Loop
    FileHandle.Close()
End Sub

A legacy FileOpen() solution:

Sub InputData()
    Dim CDTitle, CDArtist, CDLocation As String
    FileOpen(1, "MyMusic", OpenMode.Output)
    Console.WriteLine("Input CD Title: ")
    CDTitle = Console.ReadLine()
    Do Until CDTitle = "##"
        Console.WriteLine("Input CD Artist: ")
        CDArtist = Console.ReadLine()
        Console.WriteLine("Input CD Location: ")
        CDLocation = Console.ReadLine()
        Print(1, CDTitle & ":" & CDArtist & ":" & CDLocation)
        Console.WriteLine("Input CD Title: ")
        CDTitle = Console.ReadLine()
    Loop
    FileClose(1)
End Sub
5 (b): Pascal

procedure InputData;
var
  CDTitle, CDArtist, CDLocation : string;
  CDFile : Textfile;
begin
  assign(CDFile, 'MyMusic');
  rewrite(CDFile);
  writeln('Input CD Title: ');
  readln(CDTitle);
  while (CDTitle <> '##') do 
    begin
      writeln('Input CD Artist: ');
      readln(CDArtist);
      writeln('Input CD Location: ');
      readln(CDLocation);
      writeln(CDFile, CDTitle + ':' + CDArtist + ':' + CDLocation);
      writeln('Input CD Title: ');
      readln(CDTitle);
    end;
  close(CDFile);
end;

5 (b): Python

def InputData()
  :
    #CDTitle String (or CDTitle = ")
    #CDArtist String (or CDArtist = ")
    #CDLocation String (or CDLocation = ")

    FileHandle = open("MyMusic", "w")
    CDTitle = input("Input CD Title: ")
    while CDTitle != "##" :
      CDArtist = input("Input CD Artist: ")
      CDLocation = input("Input CD location: ")
      FileHandle.write(CDTitle + ";" + CDArtist + ";" + CDLocation)
      CDTitle = input("Input CD Title: ")
    FileHandle.close()