Cambridge International AS & A Level

**Cambridge Assessment International Education**
Cambridge International Advanced Subsidiary and Advanced Level

**COMPUTER SCIENCE** **9608/22**

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.



© UCLES 2018

## 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.

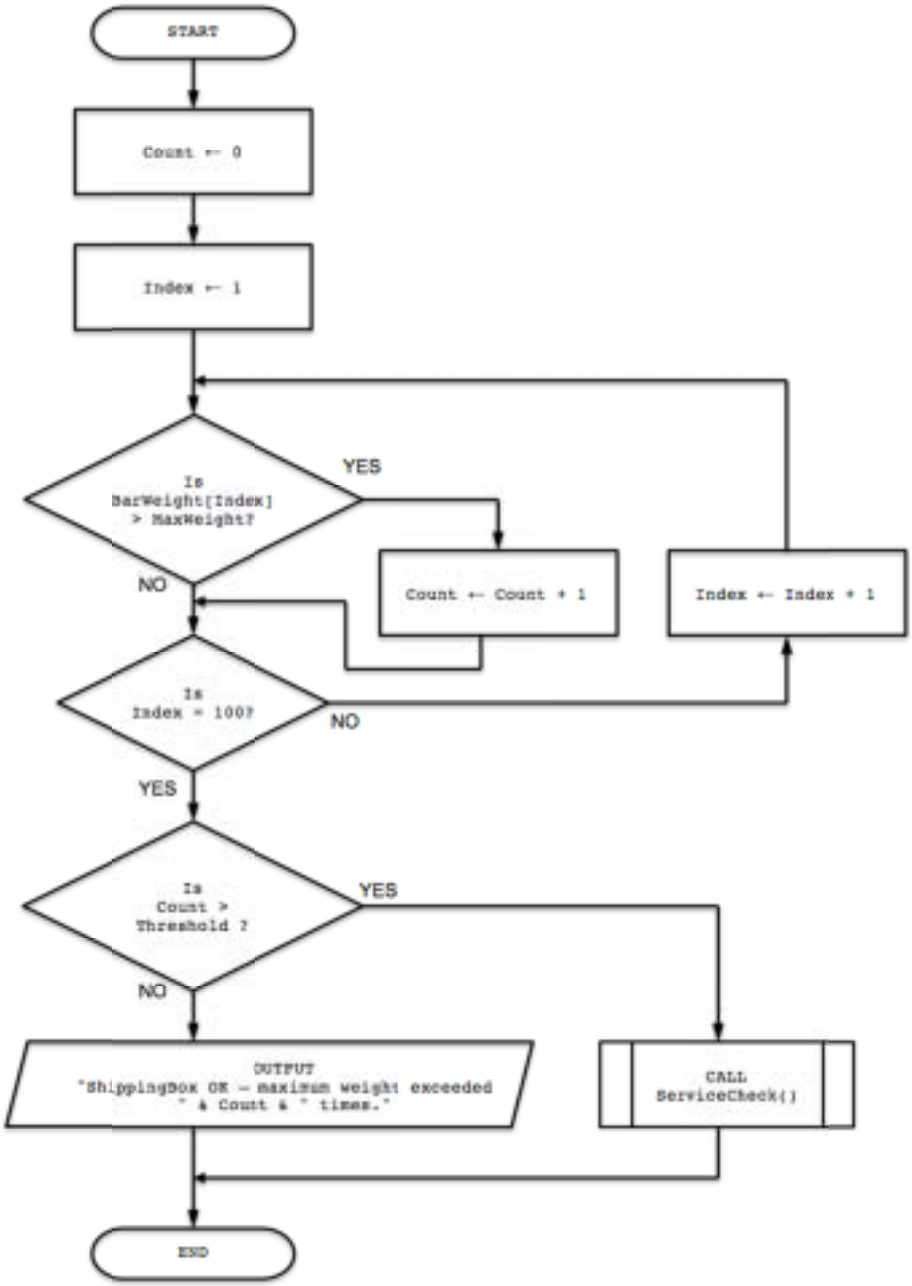
© UCLES 2018

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>1(a)</td><td>
<table>
<tr><th>Description of data item</th><th>Suitable identifier name</th></tr>
<tr><td>The temperature inside the greenhouse</td><td>GreenhouseTemperature</td></tr>
<tr><td>The temperature outside the greenhouse</td><td>OutsideTemperature</td></tr>
<tr><td>The greenhouse identification number</td><td>GreenhouseID</td></tr>
<tr><td>The time the temperature was measured</td><td>SampleTime</td></tr>
</table>
The above are examples only.<br>Names must be meaningful and unambiguous
</td><td>4</td></tr>
<tr><td>1(b)(i)</td><td>
<table>
<tr><th>Expression</th><th>Evaluates to</th></tr>
<tr><td>"Fas" & MID(Subject, 6, 3)</td><td>"Faster"</td></tr>
<tr><td>LEFT(Mark, 1)</td><td>ERROR</td></tr>
<tr><td>10 + ASC(Grade)</td><td>76</td></tr>
<tr><td>MOD(AverageMark * 2, 3)</td><td>0</td></tr>
<tr><td>CourseCompleted AND (Mark >= 60)</td><td>TRUE</td></tr>
</table>
</td><td>5</td></tr>
<tr><td>1(b)(ii)</td><td>
<table>
<tr><th>Variable</th><th>Data type</th></tr>
<tr><td>Mark</td><td>INTEGER</td></tr>
<tr><td>Subject</td><td>STRING</td></tr>
<tr><td>Grade</td><td>CHAR</td></tr>
<tr><td>CourseCompleted</td><td>BOOLEAN</td></tr>
<tr><td>AverageMark</td><td>REAL</td></tr>
</table>
One mark per answer
</td><td>5</td></tr>
</table>

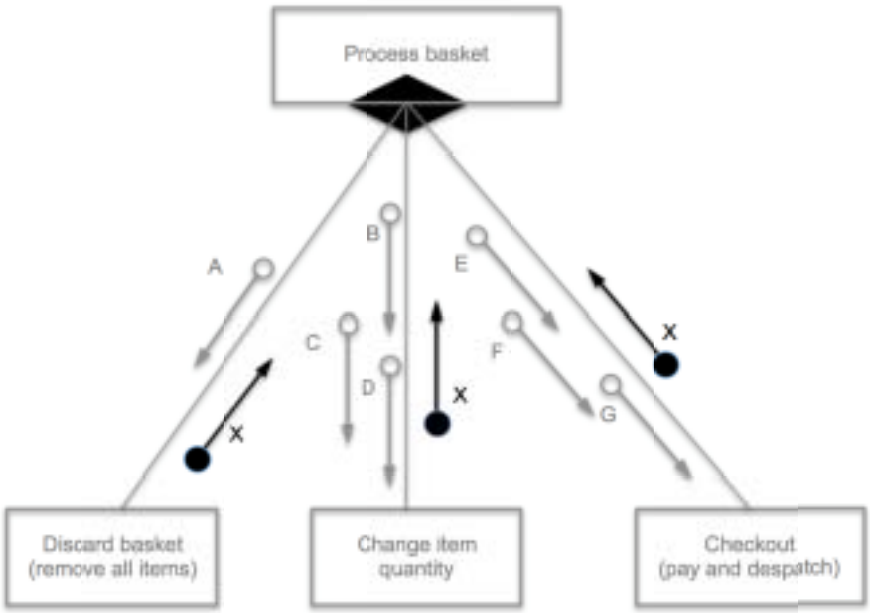
© UCLES 2018

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>2(a)(i)</td><td>• Keywords in capitals<br>• White Space / blank lines / grouping<br>• Comments<br>• Sensible function names</td><td>Max 2</td></tr>
<tr><td>2(a)(ii)</td><td>• Indentation<br>• Meaningful identifier names</td><td>2</td></tr>
<tr><td>2(b)</td><td>
<table>
<tr><th>Feature</th><th>Answer</th></tr>
<tr><td>A line number containing an example of an assignment statement</td><td>08 / 09 / 13 / 14 / 15 / 21</td></tr>
<tr><td>A line number containing the start of a ‘pre-condition’ loop</td><td>11</td></tr>
<tr><td>A line number containing the end of a ‘pre-condition’ loop</td><td>24</td></tr>
<tr><td>A line number containing the start of a selection statement</td><td>17</td></tr>
<tr><td>The number of parameters of the <code>LEFT()</code> function</td><td>2</td></tr>
<tr><td>The Boolean operator used</td><td>OR</td></tr>
<tr><td>The number of times the function <code>LEFT()</code> is called from within <code>CountDigits()</code> resulting from the call:<br><code>Result ← CountDigits("AB27C4")</code></td><td>6</td></tr>
<tr><td>The number of local variables</td><td>3</td></tr>
</table>
</td><td>8</td></tr>
<tr><td>2(c)(i)</td><td>• Mistake: function header returns a <code>CHAR</code> but last line of code returns an <code>INTEGER</code><br>• Correction: Function should return an <code>INTEGER</code> // Change line 26 to return c as <code>CHAR</code></td><td>2</td></tr>
</table>

| Question | Answer | Marks |
|---|---|---|
| 2(c)(ii) | `IF (nc >= '0') AND (nc <= '9')`<br>`  THEN`<br>`  c ← c + 1`<br>`ENDIF`<br><br>One mark for each of:<br>• Single `IF THEN ENDIF` statement (no `ELSE`)<br>• Switching `OR` to `AND`<br>• Lower value comparison<br>• Upper value comparison<br><br>ALTERNATIVE:<br><br>`IF NOT ((nc < '0') OR (nc > '9'))`<br>`  THEN`<br>`  c ← c + 1`<br>`ENDIF`<br><br>One mark for each of:<br>• Single `IF THEN ENDIF` statement (no `ELSE`)<br>• Inverting test using `NOT`…<br>• ... correct use of brackets<br>• … both comparisons unchanged | **4** |

| Question | Answer | Marks |
|---|---|---|
| 3 | Example Program Flowchart<br><br> | **10** |

| Question | Answer | Marks |
|---|---|---|
| 3 | One mark for each of:<br><br>1 START and END / STOP<br>2 Initialising `Count` to 0<br>3 Initialising `Index` to 1 or 0<br>4 Decision box comparing `BarWeight[Index]` > `MaxWeight`<br>5 Decision box comparing `Index` to 100<br>6 Decision box comparing `Count` > `Threshold`<br>7 Correct increment of `Index`<br>8 Correct increment of `Count`<br>9 Output message (concatenation of text and value) if threshold not exceeded<br>10 Calling `ServiceCheck()` if `Threshold` exceeded (without text message) | **10** |

| Question | Answer | Marks |
|---|---|---|
| 4(a)(i) | <br><br>Mark as follows:<br><br>• One mark for three arrows all labelled X (with filled circles)<br>• One mark for diamond symbol | **2** |

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>4(a)(ii)</td><td>
<table>
<tr><th>Parameter</th><th>Information</th></tr>
<tr><td>A</td><td><code>BasketID</code></td></tr>
<tr><td>B</td><td rowspan="3"><code>BasketID, ItemID, Quantity</code><br>(In any order)</td></tr>
<tr><td>C</td></tr>
<tr><td>D</td></tr>
<tr><td>E</td><td rowspan="3"><code>BasketID, DeliveryAddress, PaymentDetails</code><br>(In any order)</td></tr>
<tr><td>F</td></tr>
<tr><td>G</td></tr>
</table>
Mark as follows:<br><br>
• One mark for parameter A<br>
• One mark for parameters B, C & D<br>
• One mark for parameters E, F & G
</td><td>3</td></tr>
</table>

| Question | Answer | Marks |
|---|---|---|
| 5(a) | One mark for each of:<br><br>• Same data type using a single identifier / more efficient coding / less declaration statements needed<br>• Access of individual elements (using subscript / index)<br>• Ability to iterate through the data // easier to search / sort the data<br>• Code easier to understand / maintain / modify | **Max 2** |
| 5(b) | One mark for each of:<br><br>• (Dynamic) syntax checking / Errors are highlighted / underlined<br>• Type checking<br>• Parameter checking<br>• Identification of unused variables | **Max 2** |

© UCLES 2018

| Question | Answer | Marks |
|---|---|---|
| 5(c) | Example 'Pseudocode' solution included here for development and clarification of mark scheme.<br>Programming language example solutions appear in the Appendix.<br><br>`FUNCTION GetNumber() AS STRING`<br><br>`  DECLARE Valid : BOOLEAN`<br>`  DECLARE MemberNumber : INTEGER`<br><br>`  REPEAT`<br>`    Valid ← TRUE`<br>`    OUTPUT "Please input a valid member number"`<br>`    INPUT MemberNumber`<br>`    IF (MemberNumber > 9999) OR (MemberNumber < 1111)`<br>`      THEN`<br>`        Valid ← FALSE`<br>`    ENDIF`<br>`  UNTIL Valid = TRUE`<br><br>`  RETURN STR(MemberNumber)`<br>`ENDFUNCTION`<br><br>1 mark for each of the following:<br><br>1 Function heading (as above) and ending<br>2 Conditional loop structure<br>3 Condition to check valid membership number<br>4 Returning string value | **4** |

| Question | Answer | Marks |
|---|---|---|
| 5(d) | 'Pseudocode' solution included here for development and clarification of mark scheme.<br>Programming language solutions appear in the Appendix.<br><br>`PROCEDURE OutputLowestScore()`<br>`  DECLARE FileData, FileMembershipNumber : STRING`<br>`  DECLARE FileScore, LowestScore : INTEGER`<br>`  DECLARE LowestScoreDate, MembershipNumber : STRING`<br><br>`  MembershipNumber ← GetNumber()`<br><br>`  OPENFILE "ScoreDetails.txt" FOR READ`<br>`  LowestScore ← 100`<br><br>`  WHILE NOT EOF("ScoreDetails.txt")`<br>`  READFILE "ScoreDetails.txt", FileData`<br>`  FileMembershipNumber ← LEFT(FileData, 4)`<br>`  IF FileMembershipNumber = MembershipNumber`<br>`  THEN`<br>`  FileScore ← INT(RIGHT(FileData,2))`<br>`  IF FileScore < LowestScore`<br>`  THEN`<br>`  LowestScore ← FileScore`<br>`  LowestScoreDate ← MID(FileData(5,6))`<br>`  ENDIF`<br>`  ENDIF`<br>`  ENDWHILE`<br>`  OUTPUT ("The lowest score was " & LowestScore & " on " & _`<br>`  LowestScoreDate)`<br>`  CLOSEFILE("ScoreDetails.txt ")`<br><br>`ENDPROCEDURE`<br><br>1 mark for each of the following:<br><br>1 Declare variables to store `LowestScore` as `INTEGER` and `FileData` as `STRING` (commented in Python) (variable names may be different)<br>2 Function call `GetNumber()` for membership number<br>3 Initialisation of `LowestScore` to 99 or above<br>4 Open file in READ mode<br>5 Loop until `EOF()`<br>6 Read a line from the file **in a loop**<br>7 Use of substring operations to extract at least one data item<br>8 Compare the membership numbers<br>9 Convert score to `INTEGER`<br>10 Compare and assign (if appropriate) new value to `LowestScore`<br>11 Output the lowest score message including lowest score and date (outside the loop)<br>12 Close the file | **Max 10** |

| Question | Answer | Marks |
|---|---|---|
| 6(a) | • lower bound<br>• upper bound | **2** |
| 6(b) | Example using single temp variable:<br><br>`PROCEDURE Flip()`<br>`  //Use of Single temp value`<br><br>`  DECLARE temp : INTEGER`<br>`  DECLARE i : INTEGER  //i is the row`<br>`  DECLARE j : INTEGER  //j is the column`<br><br>`  FOR i ← 1 TO 5`<br>`    FOR j ← 1 to 4  //swap element 1&8, 2&7, 3&6, 4&5`<br>`      temp ← Picture[i,j]`<br>`      Picture[i,j] ←  Picture[i, 9 - j]`<br>`      Picture[i, 9 - j] ← temp`<br>`    ENDFOR`<br>`ENDFOR`<br><br>`ENDPROCEDURE`<br><br>Alternative Solution – Use of temp array row:<br><br>`PROCEDURE Flip()`<br>`  //Use of temproary row (8 elements)`<br><br>`  DECLARE temp : ARRAY[1:8] OF INTEGER`<br>`  DECLARE i : INTEGER  //i is the row`<br>`  DECLARE j : INTEGER  //j is the column`<br><br>`  FOR i ← 1 to 5`<br>`    FOR j ← 1 to 8`<br>`      temp[j] ← Picture[i, 9 – j]  //temp is row i reversed`<br>`    ENDFOR`<br>`    FOR j ← 1 to 8`<br>`      Picture[i,j] ← temp[j]  //copy temp back to row i`<br>`    ENDFOR`<br>`  ENDFOR`<br>`ENDPROCEDURE` | **8** |

| Question | Answer | Marks |
|---|---|---|
| 6(b) | Alternative Solution – Use of new array:<br><br>`PROCEDURE Flip()`<br>`  //Flip to New array`<br><br>`  DECLARE NewPic : ARRAY[1:5, 1:8] OF INTEGER`<br>`  DECLARE i : INTEGER  //i is the row`<br>`  DECLARE j : INTEGER  //j is the column`<br><br>`  FOR i ← 1 to 5`<br>`  FOR j ← 1 to 8`<br>`  NewPic[i, 9 - j] ← Picture[i, j]  //NewPic row is Pic row flipped`<br>`  ENDFOR`<br>`  ENDFOR`<br>`END PROCEDURE`<br><br>1 mark for each of the following (all methods):<br><br>1 Correct procedure heading and ending<br>2 Declaring local variables for loop counter(s)<br>3 Declaring a temporary storage variable for swap or new duplicate 2D array<br>4 A nested loop including attempt at flip operation …<br>5 … Correct number of iterations<br>6 Assign element to temp (single var or temp array) or to new array<br>7 Selection of correct source element (row, column)<br>8 Selection of correct destination element (row, column) | **8** |

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

© UCLES 2018

## Appendix

## Program Code Example Solutions

### Q5 (c): Visual Basic

```
Function GetNumber() As String
    Dim Valid As Boolean
    Dim MemberNumber As Integer
    Do
        Valid = True
        Console.Write("Please enter a valid member number: ")
        MemberNumber = Console.ReadLine()
        If MemberNumber > 9999 Or MemberNumber < 1111 Then
            Valid = False
        End If
    Loop Until Valid = True
    Return MemberNumber.ToString()
End Function
```

### Q5 (c): Pascal

```
function GetNumber() : string;
  var
    Valid : boolean;
    MemberNumber : integer;
  begin
     repeat
        Valid := true;
        write('Please enter a valid member number: ');
        readln(MemberNumber);
        if (MemberNumber > 9999) or (MemberNumber < 1111) then
           Valid := false;
     until Valid = true;
     GetNumber := IntToStr(MemberNumber);
  end;
```

### Q5 (c): Python

```
def GetNumber() :
    # Valid : boolean
    # MemberNumber : integer
    Valid = False
    while not Valid :
        Valid = True
        MemberNumber = int(input("Please enter a valid member number: "))
        if MemberNumber > 9999 or MemberNumber < 1111 :
            Valid = False
    return str(MemberNumber)
```

© UCLES 2018

**Q5 (d): Visual Basic**

```
Sub OutputLowestScore()
   Dim FileData As String
   Dim FileMembershipNumber As String
   Dim FileScore As Integer
   Dim LowestScore As Integer
   Dim LowestScoreDate As String
   Dim MembershipNumber As String
   MembershipNumber = GetNumber()
   FileOpen(1, "ScoreDetails.txt", OpenMode.Input)
   LowestScore = 100
   While Not EOF(1)
       FileData = LineInput(1)
       FileMembershipNumber = Left(FileData, 4)
       If FileMembershipNumber = MembershipNumber Then
           FileScore = Integer.Parse(Right(FileData, 2))
           If FileScore < LowestScore Then
               LowestScore = FileScore
               LowestScoreDate = Mid(FileData, 5, 6)
           End If
       End If
   End While
   Console.WriteLine("The lowest score was " & LowestScore & " on " &
LowestScoreDate)
   FileClose(1)
End Sub
```

© UCLES 2018

## Q5 (d): Pascal

```
procedure OutputLowestScore();
var
    FileData : string;
    FileMembershipNumber : string;
    FileScore : integer;
    LowestScore : integer;
    LowestScoreDate : string;
    MembershipNumber : string;
    Scores : textFile;
begin
    MembershipNumber := GetNumber();
    assignFile(Scores, 'ScoreDetails.txt');
    reset(Scores);
    LowestScore := 100;
    while not eof(Scores) do
    begin
        readln(Scores, FileData);
        FileMembershipNumber := copy(FileData, 1, 4);
        if FileMembershipNumber = MembershipNumber then
        begin
            FileScore:= StrToInt(copy(FileData, 11, 2));
            if FileScore < LowestScore then
            begin
                LowestScore := FileScore;
                LowestScoreDate := copy(FileData, 5, 6)
            end;
        end;
    end;
    writeln('The lowest score was ', LowestScore, ' on ', LowestScoreDate);
    close(Scores);
end;
```

© UCLES 2018

## Q5 (d): Python

```
def OutputLowestScore() :
    # FileData : string
    # FileMembershipNumber : string
    # FileScore : integer
    # LowestScore : integer
    # LowestScoreDate : string
    # MembershipNumber : string
    # File : file handle
    MembershipNumber = GetNumber()
    File = open("ScoreDetails.txt", "r")
    LowestScore = 100
    FileData = File.readline()
    while FileData != "":
        FileMembershipNumber = FileData[:4]
        if FileMembershipNumber == MembershipNumber :
            FileScore = int(FileData[10:12])
            if FileScore < LowestScore :
                LowestScore = FileScore
                LowestScoreDate = FileData[4:10
        FileData = File.readline()
    print("The lowest score was ", LowestScore, " on ", LowestScoreDate)
    File.close()
```

© UCLES 2018