

---

**COMPUTER SCIENCE**

**9608/21**

Paper 2 Written Paper

**May/June 2019**

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

---

This document consists of **11** 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)	<p><b>Construct:</b> Assignment <b>Pseudocode example:</b> Answer ← "YES"</p> <p><b>Construct:</b> Selection <b>Pseudocode example:</b> IF X = 3 THEN OUTPUT "HELLO"</p> <p><b>Construct:</b> Repetition / Iteration <b>Pseudocode example:</b> FOR N ← 1 to 100</p> <p>One mark for construct One mark for pseudocode example Maximum 4 marks</p>	<b>4</b>																				
1(a)(ii)	<table border="1" data-bbox="339 696 1289 1193"> <thead> <tr> <th data-bbox="339 696 836 761">Pseudocode statement</th> <th data-bbox="836 696 986 761">Input</th> <th data-bbox="986 696 1137 761">Process</th> <th data-bbox="1137 696 1289 761">Output</th> </tr> </thead> <tbody> <tr> <td data-bbox="339 761 836 864">Temp ← SensorValue * Factor</td> <td data-bbox="836 761 986 864"></td> <td data-bbox="986 761 1137 864" style="text-align: center;">✓</td> <td data-bbox="1137 761 1289 864"></td> </tr> <tr> <td data-bbox="339 864 836 967">WRITEFILE "LogFile.txt", TextLine</td> <td data-bbox="836 864 986 967"></td> <td data-bbox="986 864 1137 967"></td> <td data-bbox="1137 864 1289 967" style="text-align: center;">✓</td> </tr> <tr> <td data-bbox="339 967 836 1070">WRITEFILE "LogFile.txt", MyName &amp; MyIDNumber</td> <td data-bbox="836 967 986 1070"></td> <td data-bbox="986 967 1137 1070" style="text-align: center;">✓</td> <td data-bbox="1137 967 1289 1070" style="text-align: center;">✓</td> </tr> <tr> <td data-bbox="339 1070 836 1193">READFILE "AddressBook.txt", NextLine</td> <td data-bbox="836 1070 986 1193" style="text-align: center;">✓</td> <td data-bbox="986 1070 1137 1193" style="text-align: center;">(✓)</td> <td data-bbox="1137 1070 1289 1193"></td> </tr> </tbody> </table> <p data-bbox="316 1227 655 1261">One mark per correct row</p>	Pseudocode statement	Input	Process	Output	Temp ← SensorValue * Factor		✓		WRITEFILE "LogFile.txt", TextLine			✓	WRITEFILE "LogFile.txt", MyName & MyIDNumber		✓	✓	READFILE "AddressBook.txt", NextLine	✓	(✓)		<b>4</b>
Pseudocode statement	Input	Process	Output																			
Temp ← SensorValue * Factor		✓																				
WRITEFILE "LogFile.txt", TextLine			✓																			
WRITEFILE "LogFile.txt", MyName & MyIDNumber		✓	✓																			
READFILE "AddressBook.txt", NextLine	✓	(✓)																				
1(b)(i)	<table border="1" data-bbox="339 1290 1289 1686"> <thead> <tr> <th data-bbox="339 1290 1062 1355">Expression</th> <th data-bbox="1062 1290 1289 1355">Evaluates to</th> </tr> </thead> <tbody> <tr> <td data-bbox="339 1355 1062 1420">MID(Title, 5, 3) &amp; RIGHT(Author, 3)</td> <td data-bbox="1062 1355 1289 1420" style="text-align: center;"><b>"tripod"</b></td> </tr> <tr> <td data-bbox="339 1420 1062 1485">INT(WeightEach * PackSize)</td> <td data-bbox="1062 1420 1289 1485" style="text-align: center;"><b>24</b></td> </tr> <tr> <td data-bbox="339 1485 1062 1550">PackSize &gt;= 4 AND WeightEach &lt; 6.2</td> <td data-bbox="1062 1485 1289 1550" style="text-align: center;"><b>FALSE</b></td> </tr> <tr> <td data-bbox="339 1550 1062 1615">LEFT(Author, ASC(Version) - 65)</td> <td data-bbox="1062 1550 1289 1615" style="text-align: center;"><b>"Er"</b></td> </tr> <tr> <td data-bbox="339 1615 1062 1686">RIGHT(Title, (LEN(Author) - 6))</td> <td data-bbox="1062 1615 1289 1686" style="text-align: center;"><b>"hetti"</b></td> </tr> </tbody> </table> <p data-bbox="316 1720 655 1785">Quotes must be present Must be capital E in row 4</p>	Expression	Evaluates to	MID(Title, 5, 3) & RIGHT(Author, 3)	<b>"tripod"</b>	INT(WeightEach * PackSize)	<b>24</b>	PackSize >= 4 AND WeightEach < 6.2	<b>FALSE</b>	LEFT(Author, ASC(Version) - 65)	<b>"Er"</b>	RIGHT(Title, (LEN(Author) - 6))	<b>"hetti"</b>	<b>5</b>								
Expression	Evaluates to																					
MID(Title, 5, 3) & RIGHT(Author, 3)	<b>"tripod"</b>																					
INT(WeightEach * PackSize)	<b>24</b>																					
PackSize >= 4 AND WeightEach < 6.2	<b>FALSE</b>																					
LEFT(Author, ASC(Version) - 65)	<b>"Er"</b>																					
RIGHT(Title, (LEN(Author) - 6))	<b>"hetti"</b>																					

Question	Answer	Marks												
1(b)(ii)	<table border="1"> <thead> <tr> <th>Variable</th> <th>Data type</th> </tr> </thead> <tbody> <tr> <td>Tile</td> <td>STRING</td> </tr> <tr> <td>Version</td> <td>CHAR</td> </tr> <tr> <td>PackSize</td> <td>INTEGER</td> </tr> <tr> <td>WeightEach</td> <td>REAL</td> </tr> <tr> <td>Paperback</td> <td>BOOLEAN</td> </tr> </tbody> </table> <p>One mark per data type</p>	Variable	Data type	Tile	STRING	Version	CHAR	PackSize	INTEGER	WeightEach	REAL	Paperback	BOOLEAN	5
Variable	Data type													
Tile	STRING													
Version	CHAR													
PackSize	INTEGER													
WeightEach	REAL													
Paperback	BOOLEAN													
1(c)	<p>Data is chosen:</p> <ul style="list-style-type: none"> <li>to test that the program does what it is supposed to do / to check that the results are as expected</li> <li>to use known valid, boundary and erroneous values</li> </ul>	2												

Question	Answer	Marks
2(a)	<p>Type: <u>Conditional</u></p> <p>Explanation: The number of iterations is not known / dependent on a condition</p>	2
2(b)	<p>One mark per bullet point to max 3</p> <ul style="list-style-type: none"> <li>Functions / Procedures / Modules / subtasks</li> <li>Parameters</li> <li>Variable / constant declaration / assignment / Data types</li> <li>Input / Output</li> <li>Arithmetic / logic operations</li> <li>Classes / Objects</li> </ul>	3
2(c)	<p>One mark for:</p> <ul style="list-style-type: none"> <li>A <code>CASE</code> structure</li> </ul> <p>Max 2 for remaining points:</p> <ul style="list-style-type: none"> <li>Selecting on / using variable <code>X</code></li> <li>Calling <code>ProcA</code> if <code>X = 15</code></li> <li>Assigning a value of 0 to <code>Y</code> if <code>X = 20</code> <b>and</b> assign 99 to <code>Y</code> if <code>X = 25</code></li> <li>Calling <code>ProcError</code> if no match (previous conditions not satisfied) // Call <code>ProcError</code> if <code>x = NONE</code></li> </ul>	3

Question	Answer	Marks
3(a)	<pre>TotalValue ← 0 ZeroCount ← 0  FOR Index ← 1 TO 100   TotalValue ← TotalValue + Result[Index]   IF Result[Index] = 0.0     THEN       ZeroCount ← ZeroCount + 1   ENDIF ENDFOR  OUTPUT "The average is ", (TotalValue / 100) OUTPUT "The number of elements with a zero value is ",   ZeroCount</pre> <p>One mark for each of the following:</p> <ol style="list-style-type: none"> <li>1 Both initialisations</li> <li>2 Loop 100 times</li> <li>3 Adding individual element to TotalValue <b>in a loop</b></li> <li>4 Check if element value is zero <b>in a loop</b></li> <li>5 If so increment ZeroCount <b>in a loop</b></li> <li>6 Average is calculated after the loop</li> <li>7 Both OUTPUT statements, including message and variables</li> </ol>	<b>7</b>
3(b)	<pre><u>PROCEDURE ScanArray</u> (<u>BYREF AverageValue: REAL</u>, <u>BYREF ZeroCount: INTEGER</u>, <u>ArrayName : ARRAY</u>)</pre> <p>One mark for each underlined part</p> <p>Names unimportant but first two parameters must be BYREF</p>	<b>4</b>

Question	Answer	Marks																																																																																																												
4(a)(i)	<table border="1"><thead><tr><th data-bbox="368 280 523 331">Index</th><th data-bbox="523 280 746 331">AfterSpace</th><th data-bbox="746 280 938 331">NextChar</th><th data-bbox="938 280 1262 331">NewString</th></tr></thead><tbody><tr><td></td><td>FALSE</td><td></td><td>""</td></tr><tr><td>1</td><td></td><td>'X'</td><td>"X"</td></tr><tr><td>2</td><td>TRUE</td><td>'∇'</td><td>"X∇"</td></tr><tr><td>3</td><td></td><td>'∇'</td><td></td></tr><tr><td>4</td><td></td><td>'∇'</td><td></td></tr><tr><td>5</td><td>FALSE</td><td>'Y'</td><td>"X∇Y"</td></tr><tr><td>6</td><td>TRUE</td><td>'∇'</td><td>"X∇Y∇"</td></tr><tr><td>7</td><td>FALSE</td><td>'a'</td><td>"X∇Y∇a"</td></tr><tr><td>8</td><td></td><td>'n'</td><td>"X∇Y∇an"</td></tr><tr><td>9</td><td></td><td>'d'</td><td>"X∇Y∇and"</td></tr><tr><td>10</td><td>TRUE</td><td>'∇'</td><td>"X∇Y∇and∇"</td></tr><tr><td>11</td><td></td><td>'∇'</td><td></td></tr><tr><td>12</td><td>FALSE</td><td>'Z'</td><td>"X∇Y∇and∇z"</td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr></tbody></table> <p>One mark for each area as outlined. If no row marks then mark by column for columns 2, 3 and 4</p>	Index	AfterSpace	NextChar	NewString		FALSE		""	1		'X'	"X"	2	TRUE	'∇'	"X∇"	3		'∇'		4		'∇'		5	FALSE	'Y'	"X∇Y"	6	TRUE	'∇'	"X∇Y∇"	7	FALSE	'a'	"X∇Y∇a"	8		'n'	"X∇Y∇an"	9		'd'	"X∇Y∇and"	10	TRUE	'∇'	"X∇Y∇and∇"	11		'∇'		12	FALSE	'Z'	"X∇Y∇and∇z"																																																					6
Index	AfterSpace	NextChar	NewString																																																																																																											
	FALSE		""																																																																																																											
1		'X'	"X"																																																																																																											
2	TRUE	'∇'	"X∇"																																																																																																											
3		'∇'																																																																																																												
4		'∇'																																																																																																												
5	FALSE	'Y'	"X∇Y"																																																																																																											
6	TRUE	'∇'	"X∇Y∇"																																																																																																											
7	FALSE	'a'	"X∇Y∇a"																																																																																																											
8		'n'	"X∇Y∇an"																																																																																																											
9		'd'	"X∇Y∇and"																																																																																																											
10	TRUE	'∇'	"X∇Y∇and∇"																																																																																																											
11		'∇'																																																																																																												
12	FALSE	'Z'	"X∇Y∇and∇z"																																																																																																											
4(a)(ii)	To remove repeated space characters	1																																																																																																												
4(a)(iii)	<ul style="list-style-type: none"><li>• All leading spaces / spaces at the beginning</li><li>• will be removed (from the input string)</li></ul>	2																																																																																																												

Question	Answer	Marks
4(b)	<pre> DEclare Code : ARRAY[1:500, 1:4] OF STRING DEclare RowIndex : INTEGER DEclare ColIndex : INTEGER  FOR RowIndex ← 1 TO 500   FOR ColIndex ← 1 TO 4     Code[RowIndex, ColIndex] ← "Empty"   ENDFOR ENDFOR </pre> <p>One mark for each of the following:</p> <ol style="list-style-type: none"> <li>1 Array declaration</li> <li>2 Additional local variable</li> <li>3 Nested loops</li> <li>4 Array element assignment <b>within the inner loop</b></li> </ol> <p>RowIndex and ColIndex can be interchangeable</p>	<b>4</b>
4(c)	Adaptive Maintenance	<b>1</b>

Question	Answer	Marks
5(a)	<ul style="list-style-type: none"> <li>• Saves development time / no need to write it / can't write it...</li> <li>• Pre-compiled and tested / Increased reliability / reduces chance of error</li> <li>• Is available to all programs</li> </ul>	<b>3</b>
5(b)	<pre> PROCEDURE TestRand()      DECLARE MyArray : ARRAY [1:50] OF BOOLEAN     DECLARE Attempts : INTEGER     DECLARE NumFound : INTEGER     DECLARE ThisRndNumber : INTEGER     DECLARE Index : INTEGER      FOR Index ← 1 TO 50         Myarray[Index] ← FALSE     ENDFOR      NumFound ← 0     Attempts ← 0      WHILE NumFound &lt; 50         ThisRndNumber ← 1 + INT(RAND(50))         Attempts ← Attempts + 1         IF MyArray[ThisRndNumber] = FALSE             THEN                 MyArray[ThisRndNumber] ← TRUE                 NumFound ← NumFound + 1             ENDIF     ENDWHILE      OUTPUT "Number of calls to RAND() was ", Attempts  ENDPROCEDURE </pre> <p>1 mark for each of the following:</p> <ol style="list-style-type: none"> <li>1 Declaration of array of 50 elements</li> <li>2 Loop to initialise array</li> <li>3 Conditional loop stopping when all numbers generated</li> <li>4 Generate a random integer in the range 1 to 50 <b>in a loop</b></li> <li>5 Count each call to RND () <b>in a loop</b></li> <li>6 check if the number has already been generated <b>in a loop</b></li> <li>7 if true, record as generated <b>in a loop</b></li> <li>8 Output a message plus the Attempts <b>outside a loop</b></li> </ol>	<b>8</b>



Question	Answer	Marks
6(a)	One mark for each of: <ol style="list-style-type: none"> <li>1 To make a more manageable / understandable solution</li> <li>2 Subroutine may be (independently) tested and debugged</li> <li>3 Program is easier to maintain</li> </ol>	<b>3</b>
6(b)	<p>'Pseudocode' solution included here for development and clarification of mark scheme. Programming language example solutions appear in the Appendix.</p> <pre> FUNCTION FindCD(SearchCDArtist : STRING,                 SearchCDTitle : STRING) RETURNS STRING    DECLARE CDTitle : STRING   DECLARE CDArtist : STRING   DECLARE CDLocation : STRING   DECLARE Location : STRING    Location ← ""    OPENFILE "MyCDs.txt" FOR READ    WHILE NOT EOF ("MyCDs.txt") AND Location = ""      READFILE "MyCDs.txt", CDArtist     READFILE "MyCDs.txt", CDTitle     READFILE "MyCDs.txt", CDLocation      IF SearchCDArtist = CDArtist AND        SearchCDTitle = CDTitle       THEN         Location ← CDLocation       ENDIF    ENDWHILE    CLOSEFILE ("MyCDs.txt")   RETURN Location  ENDFUNCTION </pre> <p>One mark for each of the following:</p> <ol style="list-style-type: none"> <li>1 Function header and close (where appropriate), including parameters</li> <li>2 Declaration of local <code>STRING</code> variables for <code>CDArtist</code> and <code>CDTitle</code></li> <li>3 <code>OPEN</code> and <code>CLOSE</code> file for reading (Allow <code>MyCDs</code> or <code>MyCDs.txt</code>)</li> <li>4 (<code>WHILE</code>) loop checking for <code>EOF</code></li> <li>5 read three lines from file <b>in a loop</b></li> <li>6 compare search values with file values <b>in a loop...</b></li> <li>7 ...If true, set <code>Location</code> and exit loop <b>in a loop</b></li> <li>8 Return <code>Location</code></li> </ol>	<b>8</b>

**Program Code Example Solutions****Q6 (b) (i): Visual Basic**

```
Function FindCD(SearchCDArtist As String, SearchCDTitle As String) As
String
```

```
    Dim CDTitle As String
    Dim CDArtist As String
    Dim CDLocation As String
    Dim Location As String
```

```
    Location = ""
    FileOpen(1, "MyCDs.txt", OpenMode.Input)
```

```
    Do While Not EOF(1) And Location = ""
        CDArtist = LineInput(1)
        CDTitle = LineInput(1)
        CDLocation = LineInput(1)
```

```
        If SearchCDArtist = CDArtist And SearchCDTitle = CDTitle Then
            Location = CDLocation
        End If
```

```
    Loop
    FileClose(1)
```

```
EndFunction
```

**Q6 (b) (i): Python**

```
def FindCD(SearchCDArtist, SearchCDTitle):
```

```
    # CDTitle, CDArtist, CDLocation, Location : string
```

```
    Location = ""
    myFile = open("MyCDs.txt", 'r')
    while True:
        # or Location == "":
        CDArtist = myFile.readline()
```

```
        if CDArtist == "":
            break
        else:
```

```
            CDTitle = myFile.readline()
            CDLocation = myFile.readline()
```

```
            if SearchCDArtist == CDArtist.strip() and SearchCDTitle ==
                CDTitle.strip():
```

```
                Location = CDLocation
```

```
    myFile.close
    return (Location)
```

**Q6 (b) (i): Pascal**

```
function FindCD(SearchCDArtist, SearchCDTitle:string): string;

var
  CDTitle, CDArtist, CDLocation, Location : string;
  FileHandle : TextFile;

begin
  Location := '';
  AssignFile(FileHandle, 'MyCDs.txt');
  Reset (FileHandle);

  while not eof(FileHandle) and (Location = '') do
  begin
    readln(FileHandle, CDArtist);
    readln(FileHandle, CDTitle);
    readln(FileHandle, CDLocation);
    if (SearchCDArtist = CDArtist) and (SearchCDTitle = CDTitle)
then
      Location := CDLocation;
  end;

  Close (FileHandle);
  FindCD := Location;

end;
```