

---

**COMPUTER SCIENCE**

**9608/22**

Paper 2 Written Paper

**October/November 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 October/November 2018 series for most Cambridge IGCSE™, Cambridge International A and AS Level components and some Cambridge O Level components.

---

This document consists of **13** 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)	<table border="1"> <thead> <tr> <th data-bbox="320 253 762 349">Statement</th> <th data-bbox="767 253 932 349">Selection</th> <th data-bbox="936 253 1107 349">Repetition (Iteration)</th> <th data-bbox="1112 253 1313 349">Assignment</th> </tr> </thead> <tbody> <tr> <td data-bbox="320 356 762 421">Index ← Index + 5</td> <td data-bbox="767 356 932 421"></td> <td data-bbox="936 356 1107 421"></td> <td data-bbox="1112 356 1313 421">✓</td> </tr> <tr> <td data-bbox="320 427 762 492">FOR Count ← 1 TO 100</td> <td data-bbox="767 427 932 492"></td> <td data-bbox="936 427 1107 492">✓</td> <td data-bbox="1112 427 1313 492">(✓)</td> </tr> <tr> <td data-bbox="320 499 762 595">TempValue[Index] ← ReadValue(SensorID)</td> <td data-bbox="767 499 932 595"></td> <td data-bbox="936 499 1107 595"></td> <td data-bbox="1112 499 1313 595">✓</td> </tr> <tr> <td data-bbox="320 602 762 667">IF Index &lt; 30</td> <td data-bbox="767 602 932 667">✓</td> <td data-bbox="936 602 1107 667"></td> <td data-bbox="1112 602 1313 667"></td> </tr> <tr> <td data-bbox="320 674 762 739">UNTIL DayNumber &gt; 7</td> <td data-bbox="767 674 932 739"></td> <td data-bbox="936 674 1107 739">✓</td> <td data-bbox="1112 674 1313 739"></td> </tr> <tr> <td data-bbox="320 745 762 819">OTHERWISE OUTPUT "ERROR"</td> <td data-bbox="767 745 932 819">✓</td> <td data-bbox="936 745 1107 819"></td> <td data-bbox="1112 745 1313 819"></td> </tr> </tbody> </table> <p data-bbox="316 853 1090 920">1 mark per correct row Ignore any tick in assignment column for second statement</p>				Statement	Selection	Repetition (Iteration)	Assignment	Index ← Index + 5			✓	FOR Count ← 1 TO 100		✓	(✓)	TempValue[Index] ← ReadValue(SensorID)			✓	IF Index < 30	✓			UNTIL DayNumber > 7		✓		OTHERWISE OUTPUT "ERROR"	✓			6
Statement	Selection	Repetition (Iteration)	Assignment																														
Index ← Index + 5			✓																														
FOR Count ← 1 TO 100		✓	(✓)																														
TempValue[Index] ← ReadValue(SensorID)			✓																														
IF Index < 30	✓																																
UNTIL DayNumber > 7		✓																															
OTHERWISE OUTPUT "ERROR"	✓																																
1(b)(i)	<table border="1"> <thead> <tr> <th data-bbox="320 958 1062 1023">Statement</th> <th data-bbox="1067 958 1313 1023">Data type</th> </tr> </thead> <tbody> <tr> <td data-bbox="320 1030 1062 1095">Revision ← 'B'</td> <td data-bbox="1067 1030 1313 1095">CHAR</td> </tr> <tr> <td data-bbox="320 1102 1062 1167">MaxValue ← 13.3</td> <td data-bbox="1067 1102 1313 1167">REAL</td> </tr> <tr> <td data-bbox="320 1173 1062 1238">ArrayFull ← TRUE</td> <td data-bbox="1067 1173 1313 1238">BOOLEAN</td> </tr> <tr> <td data-bbox="320 1245 1062 1310">Activity ← "Design"</td> <td data-bbox="1067 1245 1313 1310">STRING</td> </tr> <tr> <td data-bbox="320 1317 1062 1382">NumberOfEdits ← 270</td> <td data-bbox="1067 1317 1313 1382">INTEGER</td> </tr> </tbody> </table> <p data-bbox="316 1406 616 1440">1 mark per correct row</p>		Statement	Data type	Revision ← 'B'	CHAR	MaxValue ← 13.3	REAL	ArrayFull ← TRUE	BOOLEAN	Activity ← "Design"	STRING	NumberOfEdits ← 270	INTEGER		5																	
Statement	Data type																																
Revision ← 'B'	CHAR																																
MaxValue ← 13.3	REAL																																
ArrayFull ← TRUE	BOOLEAN																																
Activity ← "Design"	STRING																																
NumberOfEdits ← 270	INTEGER																																
1(b)(ii)	<table border="1"> <thead> <tr> <th data-bbox="320 1473 1062 1538">Expression</th> <th data-bbox="1067 1473 1313 1538">Evaluates to</th> </tr> </thead> <tbody> <tr> <td data-bbox="320 1545 1062 1610">MID(Activity, 3, 4) &amp; "ature"</td> <td data-bbox="1067 1545 1313 1610">"signature"</td> </tr> <tr> <td data-bbox="320 1617 1062 1682">INT(MaxValue * 2)</td> <td data-bbox="1067 1617 1313 1682">26</td> </tr> <tr> <td data-bbox="320 1688 1062 1753">ArrayFull AND NumberOfEdits &lt; 300</td> <td data-bbox="1067 1688 1313 1753">TRUE</td> </tr> <tr> <td data-bbox="320 1760 1062 1825">ASC(Revision + 1)</td> <td data-bbox="1067 1760 1313 1825">ERROR</td> </tr> <tr> <td data-bbox="320 1832 1062 1897">Activity = "Testing" OR Revision = 'A'</td> <td data-bbox="1067 1832 1313 1897">FALSE</td> </tr> </tbody> </table> <p data-bbox="316 1904 616 1937">1 mark per correct row</p>		Expression	Evaluates to	MID(Activity, 3, 4) & "ature"	"signature"	INT(MaxValue * 2)	26	ArrayFull AND NumberOfEdits < 300	TRUE	ASC(Revision + 1)	ERROR	Activity = "Testing" OR Revision = 'A'	FALSE		5																	
Expression	Evaluates to																																
MID(Activity, 3, 4) & "ature"	"signature"																																
INT(MaxValue * 2)	26																																
ArrayFull AND NumberOfEdits < 300	TRUE																																
ASC(Revision + 1)	ERROR																																
Activity = "Testing" OR Revision = 'A'	FALSE																																

Question	Answer	Marks
2(a)(i)	<pre> FUNCTION CalcPoints(CardNum: STRING, Total: REAL)                                 RETURNS INTEGER    DECLARE OldPoints : INTEGER   DECLARE NewPoints : INTEGER    IF Total &gt; 100     THEN       OldPoints ← GetPoints(CardNum)       IF OldPoints &gt; 2000         THEN           NewPoints ← INT(Total * 1.2)         ELSE           NewPoints ← INT(Total * 1.1)        ENDIF     ELSE       NewPoints ← 0     ENDIF    RETURN NewPoints  ENDFUNCTION </pre> <p>1 mark for each of the following:</p> <ol style="list-style-type: none"> <li>1 Correct FUNCTION heading (as given) and end</li> <li>2 Declaring local variables for OldPoints and NewPoints</li> <li>3 IF...THEN...ELSE...ENDIF with Total &gt; 100</li> <li>4 ...Newpoints set to zero if Total &lt;= 100</li> <li>5 Nested IF...THEN...ELSE...ENDIF with OldPoints &gt; 2000</li> <li>6 ... with correct assignments of NewPoints</li> <li>7 Return NewPoints for all cases</li> </ol>	<b>7</b>

Question	Answer	Marks
2(a)(ii)	<pre> FUNCTION GetTotal() RETURNS REAL    DECLARE Valid : BOOLEAN   DECLARE Amount : REAL   Valid ← FALSE    REPEAT     OUTPUT "Enter the amount"     INPUT Amount     IF Amount &gt; 0 AND Amount &lt; 10000       THEN         Valid ← TRUE       ENDF   UNTIL Valid = TRUE    RETURN Amount  ENDFUNCTION </pre> <p>Note that the pseudocode shown is only an example. The use of an explicit flag and IF structure are not essential provided the functionality is provided.</p> <p>1 mark for each of:</p> <ol style="list-style-type: none"> <li>1 declaration of local variable(s) used</li> <li>2 prompt followed by input</li> <li>3 conditional loop</li> <li>4 checking that input value &gt; 0 and input value &lt; 10000</li> <li>5 returning the value</li> </ol>	<b>5</b>
2(b)(i)	<p>1 mark for name, 1 mark for description Accept by example for description</p> <p>Name: Run-time Description: The program executes an illegal instruction // performs an illegal operation that is trapped by the OS</p>	<b>2</b>
2(b)(ii)	<p>One <b>specific value</b> from within each of the following ranges:</p> <p>For example:</p> <ul style="list-style-type: none"> <li>• 73                   (a value ≤ 100)</li> <li>• 145.3               (a value &gt; 100)</li> </ul> <p>1 mark for the value, plus one for a meaningful description.</p>	<b>4</b>

Question	Answer	Marks
3(a)	You should be able to recognise / understand in another language: <ul style="list-style-type: none"> <li>Declaration / assignment / sequence / selection / repetition (iteration) / Subroutines / Parameters passed between modules / program structure / Input and Output</li> </ul> <p>Any 2 marks from the list. Max 1 mark if no explanation given</p>	2
3(b)	Key points: <ul style="list-style-type: none"> <li>to increase the level of detail of the algorithm // break the problem into smaller steps</li> <li>until steps are easier to solve // to be directly translated into lines of code</li> </ul>	2
3(c)	Key points: <ul style="list-style-type: none"> <li>A <u>loop / repetition / iteration</u> to check every element</li> <li>Compare the array element with the value being searched</li> <li>Exit loop / stop search when value found or end of array reached</li> <li>If value is found then output the index position, otherwise output "Not found"</li> </ul>	4

Question	Answer	Marks										
4(a)(i)	<table border="1"> <tbody> <tr> <td>The identifier name of a local variable</td> <td>FileData / FileLine</td> </tr> <tr> <td>The identifier name of a user-defined procedure</td> <td>ScanCompleted</td> </tr> <tr> <td>The identifier name of a user-defined function</td> <td>ReadFileLine / ScanFile</td> </tr> <tr> <td>The number of dimensions of ResultArray</td> <td>1</td> </tr> <tr> <td>The scope of FileData</td> <td>Local</td> </tr> </tbody> </table> <p>1 mark for each correct answer</p>	The identifier name of a local variable	FileData / FileLine	The identifier name of a user-defined procedure	ScanCompleted	The identifier name of a user-defined function	ReadFileLine / ScanFile	The number of dimensions of ResultArray	1	The scope of FileData	Local	5
The identifier name of a local variable	FileData / FileLine											
The identifier name of a user-defined procedure	ScanCompleted											
The identifier name of a user-defined function	ReadFileLine / ScanFile											
The number of dimensions of ResultArray	1											
The scope of FileData	Local											
4(a)(ii)	Example mark points, max 4 marks: <ul style="list-style-type: none"> <li>If FileData is not empty, start and continue a loop // while FileData is not empty...</li> <li>Compare the Left 7 characters of FileData with SearchString</li> <li>If they match, add FileData to the array / ResultArray[]</li> <li>If they match, increment the array index variable</li> <li>Increment FileLine</li> <li>Read the next line from the file</li> </ul>	4										

Question	Answer	Marks
4(b)	<p>Pseudocode solution included here for development and clarification of mark scheme. Programming language solutions appear at the end of this mark scheme.</p> <pre> FUNCTION ScanFile(SearchString STRING) RETURNS INTEGER      DECLARE FileData : STRING     DECLARE FileLine : INTEGER     DECLARE NewData : STRING     DECLARE Size : INTEGER      NextArrayElement ← 1     FileLine ← 1      FileData ← ReadFileLine("DataFile.txt", FileLine)      WHILE FileData &lt;&gt; ""         IF LEFT(FileData, 7) = SearchString             THEN                 Size ← LENGTH(FileData)                 NewData ← RIGHT(FileData, Size-7)                 ResultArray[NextArrayElement] ← NewData                 NextArrayElement ← NextArrayElement + 1             ENDIF             FileLine ← FileLine + 1             FileData ← ReadFileLine("DataFile.txt", FileLine)         ENDWHILE          CALL ScanCompleted()         RETURN FileLine      ENDFUNCTION </pre> <p>1 mark for each of the following:</p> <ol style="list-style-type: none"> <li>1 Function heading and ending including parameters <b>and</b> return statement</li> <li>2 Local variable declarations for FileData <b>and</b> FileLine but <b>NOT</b> declaration of NextArrayElement</li> <li>3 initialisation of FileLine <b>and</b> NextArrayElement (allow 0 or 1)</li> <li>4 WHILE ... ENDWHILE loop</li> <li>5 Comparing SearchString with first seven characters of FileData</li> <li>6 Use of substring function <b>and</b> subtraction of 7 from length</li> <li>7 Assign NewData to array ResultArray following attempt at MP6</li> <li>8 Increment NextArrayElement <b>and</b> FileLine (as above)</li> </ol>	8
4(c)(i)	Through the use of: Subroutines / Functions / Procedures / Parameters / Methods	1
4(c)(ii)	Reduces program complexity // easier to develop / test / debug // tasks may be re-used // tasks can be allocated to different programmers/teams (with different skills) // limited scope of local variables	1

Question	Answer	Marks
4(d)	<p>Pseudocode solution included here for development and clarification of mark scheme. Programming language solutions appear at the end of this mark scheme.</p> <pre>DECLARE ResultArray : ARRAY [1:100] OF STRING DECLARE Index: INTEGER  FOR Index ← 1 TO 100     ResultArray[Index] ← "NO DATA" ENDFOR</pre> <p>1 mark for each of the following:</p> <ul style="list-style-type: none"><li>• ResultArray declaration / commented in Python</li><li>• Loop for 100 elements</li><li>• ...assignment of string "NO DATA" to indexed array element</li></ul>	<b>3</b>



Question	Answer	Marks
5	<pre> PROCEDURE LineNumber(FileName: STRING,                     StartNumber: INTEGER, StepNumber: INTEGER)      DECLARE FileData : STRING     DECLARE Count : INTEGER     DECLARE Reply : CHAR     DECLARE Continue : BOOLEAN      Count ← 0     Continue ← TRUE      OPENFILE FileName FOR READ      WHILE NOT EOF(FileName) AND Continue = TRUE          READFILE FileName, FileData         FileData ← NUM_TO_STRING(StartNumber) &amp; ": " &amp;                     FileData          OUTPUT FileData         Count ← Count + 1          IF Count = 20 THEN             OUTPUT "Do you wish to continue?"             INPUT Reply             IF Reply = 'N'                 THEN                     Continue ← FALSE                 ELSE                     Count ← 0             ENDIF         ENDIF         StartNumber ← StartNumber + StepNumber      ENDWHILE      CLOSEFILE FileName ENDPROCEDURE </pre>	11

Question	Answer	Marks
5	1 mark for each of the following to max 11:  1 Procedure heading including parameters 2 Declare an integer variable for the count 3 Open file in <code>READ</code> mode 4 Initialise <code>Count</code> variable before loop <b>and</b> increment in the loop (or other mechanism) 5 Loop including until <code>EOF(FileName)</code> (see note) 6 Call <code>READFILE()</code> <b>in a loop</b> 7 Convert <code>StartNumber</code> to string <b>in a loop</b> 8 Output concatenated string <b>in a loop</b> 9 Check if count = 20 10 Prompt and Input (inside an IF) 11 If user input = 'N' terminate loop 12 Add <code>StepNumber</code> to <code>StartNumber</code> 13 Close file	

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

**Program Code Solutions****Q4 (b): Visual Basic**

```
Function ScanFile(SearchString As String) As Integer

    Dim FileData As String
    Dim FileLine As Integer
    Dim NewData As String
    Dim Size As Integer

    NextArrayElement = 1
    FileLine = 1

    FileData = ReadFileLine("DataFile.txt", FileLine)

    Do While FileData <> ""
        If Left(FileData, 7) = SearchString Then
            Size = Len(FileData)
            NewData = Right(FileData, Size-7) // NewData = Mid(FileData, 8,
                                                                    Size-7)

            ResultArray(NextArrayElement) = NewData
            NextArrayElement = NextArrayElement + 1
        End If
        FileLine = FileLine + 1
        FileData = ReadFileLine("DataFile.txt", FileLine)
    Loop

    Call ScanCompleted() ' Keyword "Call" OK but not required
    Return FileLine

End Function
```

**Q4 (b): Pascal**

```

function ScanFile(SearchString : string) : integer;
var
  FileData : string;
  FileLine : integer;
  NewData : string;
  Size : integer;

begin
  NextArrayElement := 1;
  FileLine := 1;

  FileData := ReadFileLine('DataFile.txt', FileLine);

  while FileData <> '' do
  begin
    if leftstr(FileData, 7) = SearchString then
    begin
      Size := Length(FileData);
      NewData := rightstr(FileData, Size-7);
      ResultArray[NextArrayElement] := NewData;
      NextArrayElement := NextArrayElement + 1;
    end;
    FileLine := FileLine + 1;
    FileData := ReadFileLine("DataFile.txt", FileLine);
  end;

  ScanCompleted(); // Keyword "Call" not valid
  Return FileLine; // ScanFile := FileLine;
end;

```

**Q4 (b): Python**

```

def scanfile(searchstring):

    # filedata : string
    # fileline : integer
    # newdata : string

    nextarrayelement = 1
    fileline = 1

    filedata = readfileline("datafile.txt", fileline)

    while filedata != "" :
        if filedata[:7] == searchstring:
            newdata = filedata[7:]
            resultarray[nextarrayelement] = newdata
            nextarrayelement = nextarrayelement + 1
            fileline = fileline + 1
            filedata = readfileline("datafile.txt", fileline)

    scancompleted() # keyword "call" not valid ???
    return fileline

```

**Q4 (d): Visual Basic**

```
Dim ResultArray(99) As String
Dim Index As Integer
For Index = 0 To 99
    ResultArray(Index) = "NO DATA"
Next Index
```

**Q4 (d): Pascal**

```
var
    ResultArray : array [1..100] of string;
    Index : integer;

begin
    for Index := 1 to 100 do
        ResultArray[Index] := 'NO DATA';
    end.
end.
```

**Q4 (d): Python – alternative 1 of n**

```
#ResultArray[] as STRING
Resultarray = ["NO DATA" for index in range(100)]
```

**Q4 (d): Python – alternative 2 of n**

```
#ResultArray[] as STRING
ResultArray = []
For Index in range(100):
    ResultArray.append("NO DATA")
```

**Q4 (d): Python – alternative 3 of n**

```
# ResultArray[99] As String
ResultArray = ["NO DATA"]*100
```