
COMPUTER SCIENCE

9608/22

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 **15** 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)	<ul style="list-style-type: none"> a sequence of steps / stages / instructions to implement a task // solution to a problem <p>Allow alternatives to sequence providing meaning is clear.</p>	2												
1(a)(ii)	<p>Input:</p> <ul style="list-style-type: none"> e.g. <code>INPUT MyVar // READFILE MyFile, MyString</code> <p>Process:</p> <ul style="list-style-type: none"> e.g. <code>NextChar ← 'X' // Count ← Count + 1</code> <p>Output:</p> <ul style="list-style-type: none"> e.g. <code>OUTPUT "Hello World" // WRITEFILE "LogFile.txt", SomeData</code> <p>Mark as follows:</p> <p>One mark for each stage (Input and Process) One mark for each pseudocode example</p>	5												
1(b)(i)	<table border="1" data-bbox="261 898 1369 1290"> <thead> <tr> <th data-bbox="261 898 1043 963">Expression</th> <th data-bbox="1043 898 1369 963">Evaluates to</th> </tr> </thead> <tbody> <tr> <td data-bbox="261 963 1043 1028"><code>STRING_TO_NUM(RIGHT(ID, 3))</code></td> <td data-bbox="1043 963 1369 1028">234.0 / 234</td> </tr> <tr> <td data-bbox="261 1028 1043 1093"><code>INT(Height * Children)</code></td> <td data-bbox="1043 1028 1369 1093">11</td> </tr> <tr> <td data-bbox="261 1093 1043 1158"><code>IsMarried AND Married < 31/12/1999</code></td> <td data-bbox="1043 1093 1369 1158">TRUE</td> </tr> <tr> <td data-bbox="261 1158 1043 1223"><code>LENGTH(ID & NUM_TO_STRING(Height))</code></td> <td data-bbox="1043 1158 1369 1223">8</td> </tr> <tr> <td data-bbox="261 1223 1043 1290"><code>MID((ID, INT(Height) - Children, 2)</code></td> <td data-bbox="1043 1223 1369 1290">"23"</td> </tr> </tbody> </table> <p>No quotes for row 1 Quotes (single or double) for row 5</p>	Expression	Evaluates to	<code>STRING_TO_NUM(RIGHT(ID, 3))</code>	234.0 / 234	<code>INT(Height * Children)</code>	11	<code>IsMarried AND Married < 31/12/1999</code>	TRUE	<code>LENGTH(ID & NUM_TO_STRING(Height))</code>	8	<code>MID((ID, INT(Height) - Children, 2)</code>	"23"	5
Expression	Evaluates to													
<code>STRING_TO_NUM(RIGHT(ID, 3))</code>	234.0 / 234													
<code>INT(Height * Children)</code>	11													
<code>IsMarried AND Married < 31/12/1999</code>	TRUE													
<code>LENGTH(ID & NUM_TO_STRING(Height))</code>	8													
<code>MID((ID, INT(Height) - Children, 2)</code>	"23"													
1(b)(ii)	<table border="1" data-bbox="261 1422 1369 1814"> <thead> <tr> <th data-bbox="261 1422 588 1487">Variable</th> <th data-bbox="588 1422 1369 1487">Data type</th> </tr> </thead> <tbody> <tr> <td data-bbox="261 1487 588 1552">Married</td> <td data-bbox="588 1487 1369 1552">DATE</td> </tr> <tr> <td data-bbox="261 1552 588 1617">ID</td> <td data-bbox="588 1552 1369 1617">STRING</td> </tr> <tr> <td data-bbox="261 1617 588 1682">MiddleInitial</td> <td data-bbox="588 1617 1369 1682">CHAR</td> </tr> <tr> <td data-bbox="261 1682 588 1747">Height</td> <td data-bbox="588 1682 1369 1747">REAL</td> </tr> <tr> <td data-bbox="261 1747 588 1814">IsMarried</td> <td data-bbox="588 1747 1369 1814">BOOLEAN</td> </tr> </tbody> </table> <p>One mark per data type</p>	Variable	Data type	Married	DATE	ID	STRING	MiddleInitial	CHAR	Height	REAL	IsMarried	BOOLEAN	5
Variable	Data type													
Married	DATE													
ID	STRING													
MiddleInitial	CHAR													
Height	REAL													
IsMarried	BOOLEAN													

Question	Answer	Marks
2(a)(i)	<ul style="list-style-type: none"> • To make a more manageable / understandable solution • To support modular design 	1
2(a)(ii)	<ul style="list-style-type: none"> • Allows the subroutine to be called from many / multiple places • Subroutine may be (independently) tested and debugged • If the task changes the change needs to be made only once • Reduces unnecessary duplication / program lines • Allows teams to work on different parts of the solution 	3
2(a)(iii)	<p>Type of subroutine: Function Justification: It returns a value // assigns a value to variable <i>Answer</i></p> <p>One mark for type One mark for justification</p>	2
2(b)	<ul style="list-style-type: none"> • An editor is used to produce / write / modify the <u>source code / program / high-level language code</u> <p>OR by example:</p> <p>An editor provides (features such as) context-sensitive prompts / dynamic syntax checking etc.</p> <ul style="list-style-type: none"> • A translator (compiler) is used to translate / convert the source code / program / high-level language code into <u>object code / machine code / an executable file</u>. <p>OR</p> <p>A translator (interpreter) is used to translate the source code / program / high-level language code <u>line by line</u></p> <ul style="list-style-type: none"> • A debugger is used to test the program / detect errors (and correct errors) in the program. <p>One mark per bullet point</p>	3

Question	Answer	Marks
2(c)	<p>Control structure: A (pre-) <u>conditional</u> loop</p> <p>Function of code:</p> <ul style="list-style-type: none"> • Check if <code>Result</code> is less than 20 and If true, calls <code>ResetSensor</code> with parameter value 3... • ... and assign the value returned by <code>GetSensor</code> with parameter value 3 to <code>Result</code> • Loop until <code>Result >= 20</code> <p>OR</p> <p>Control structure: A selection // conditional statement</p> <p>Function of code:</p> <ul style="list-style-type: none"> • Check if <code>Result</code> is less than 20 and If true, calls <code>ResetSensor</code> with parameter value 3... • ... and assign the value returned by <code>GetSensor</code> with parameter value 3 to <code>Result</code> <p>One mark for control structure, maximum two for function</p> <p>Function of code marks independent of answer to control structure</p>	3

Question	Answer	Marks
3(a)(i)	<p><u>PROCEDURE SubA (A : STRING, B : INTEGER, BYREF C : CHAR)</u></p> <p>One mark for each underlined part Ignore <code>BYVAL</code> for parameter A and/or parameter B Parameter order / names not important but must be correct data types</p>	3
3(a)(ii)	<p><u>Function SubB (D : STRING, E : INTEGER) RETURNS BOOLEAN</u></p> <p>One mark for each underlined part Ignore <code>BYVAL</code> for parameter D and/or parameter E Parameter order / names not important but must be correct data types</p>	3
3(b)	<ul style="list-style-type: none"> • Selection • Iteration • Sequence <p>One mark per bullet to max. 2</p>	2

Question	Answer					Marks
4(a)(i)	Index	NextChar	Selected	NewValue	NewString	5
			0		"0"	
	1	'1'			"01"	
	2	'2'			"012"	
	3	'∇'		12		
			12			
					"0"	
	4	'3'			"03"	
	5	'4'			"034"	
	6	'∇'		34		
			34			
					"0"	
	7	'5'			"05"	
	8	'∇'		5		
					"0"	
	9	'∇'		0		
					"0"	
	10	'3'			"03"	
	11	'9'			"039"	
<p>One mark for each column.</p> <p>If no mark for columns, award one mark for initialisation of <code>Selected</code> to 0 and <code>Newstring</code> to '0' (single or double quotes).</p>						
4(a)(ii)	34					1

Question	Answer	Marks
4(b)(i)	<ul style="list-style-type: none"> • The final value (in the string) is the largest value (39) and is not considered // the final comparison with variable <code>Selected</code> is not made • The loop terminates at the end of the string (the character 9) // there wasn't a final space / non-numeric digit <p>One mark per bullet.</p>	2
4(b)(ii)	<ul style="list-style-type: none"> • Check the (final) value of <code>NewString</code> after the loop... • ...and see if it is greater than <code>Selected</code> (repeat the existing conditional clause) <p>OR</p> <ul style="list-style-type: none"> • Amend the algorithm to add a space character / non-numeric character to the end of the string... • ...before the <code>FOR</code> loop / at the start of the function <p>One mark per bullet point Accept alternative workable solution</p>	2

Question	Answer	Marks
5(a)	<p>One mark for each of:</p> <ul style="list-style-type: none"> • 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 <p>Maximum 3 marks</p>	3
5(b)	<pre>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 ENDPROCEDURE</pre> <p>One mark for each of the following:</p> <ol style="list-style-type: none"> 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 	6

Question	Answer	Marks
6(a)	<p>'Pseudocode' solution included here for development and clarification of mark scheme.</p> <p>Programming language example solutions appear in the Appendix.</p> <pre> FUNCTION GetInfo() RETURNS STRING DECLARE ID : STRING DECLARE PreferredName : STRING DECLARE Valid : BOOLEAN Valid ← FALSE WHILE Valid = FALSE OUTPUT "Please Enter a valid ID" INPUT ID IF LENGTH(ID) = 5 AND LEFT(ID, 1) >= 'A' AND LEFT(ID, 1) <= 'Z' AND ISNUM(RIGHT(ID, 4)) THEN Valid ← TRUE ENDIF ENDWHILE OUTPUT "Please enter preferred name" INPUT PreferredName RETURN ID & '*' & PreferredName ENDFUNCTION </pre> <p>One mark for each of the following:</p> <ol style="list-style-type: none"> 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 	8

Question	Answer	Marks
6(b)	<p>'Pseudocode' solution included here for development and clarification of mark scheme.</p> <p>Programming language example solutions appear in the Appendix.</p> <pre> 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 </pre> <p>One mark for each of the following:</p> <ol style="list-style-type: none"> 1. Procedure header and end 2. Conditional loop terminated with user input 3. call to <code>GetInfo()</code> in a loop 4. check first character of returned <code>UserID</code> value in a loop 5. call(s) to <code>WriteInfo()</code> in both cases ... 6. ... with two <code>STRING</code> parameters in a loop 7. exit procedure if <code>WriteInfo()</code> unsuccessful in a loop 8. if <code>WriteInfo()</code> successful, prompt and check input to repeat / exit in a loop 	8
6(c)	<pre> <u>FUNCTION WriteInfo (FileData : STRING, Filename : STRING)</u> <u>RETURNS BOOLEAN</u> </pre> <p>One mark per underlined section</p>	3

*** 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.WriteLine("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.WriteLine("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"))
Then
            Valid = True
            For i = 2 To 5
                Number = Mid(ID, i, 1)
                If (Number < "0") Or (Number > "9") Then
                    Valid = False
                End If
            Next
        End If
    End While

    Console.WriteLine("Please enter preferred name")
    PreferredName = Console.ReadLine()
    Return (ID & "*" & PreferredName)
End Function

```

Q6 (a): 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;
end;
```

Free 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;
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;
    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) ")
```