Cambridge Assessment International Education
Cambridge International Advanced Subsidiary and Advanced Level

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.

Cambridge Assessment International Education

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)	**Construct**: Assignment **Pseudocode example**: `Answer ← "YES"` **Construct**: Selection **Pseudocode example**: `IF X = 3 THEN OUTPUT "HELLO"` **Construct**: Repetition / Iteration **Pseudocode example**: `FOR N ← 1 to 100` One mark for construct One mark for pseudocode example Maximum 4 marks	**4**

1(a)(ii)

Pseudocode statement	Input	Process	Output
`Temp ← SensorValue * Factor`		✓	
`WRITEFILE "LogFile.txt", TextLine`			✓
`WRITEFILE "LogFile.txt", MyName & MyIDNumber`		✓	✓
`READFILE "AddressBook.txt", NextLine`	✓	(✓)	

One mark per correct row

Marks: **4**

1(b)(i)

Expression	Evaluates to
`MID(Title, 5, 3) & RIGHT(Author, 3)`	**"tripod"**
`INT(WeightEach * PackSize)`	**24**
`PackSize >= 4 AND WeightEach < 6.2`	**FALSE**
`LEFT(Author, ASC(Version) - 65)`	**"Er"**
`RIGHT(Title, (LEN(Author) - 6))`	**"hetti"**

Quotes must be present
Must be capital E in row 4

Marks: **5**

Question	Answer	Marks
1(b)(ii)	<table><tr><td>**Variable**</td><td>**Data type**</td></tr><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></table> One mark per data type	5
1(c)	Data is chosen: • to test that the program does what it is supposed to do / to check that the results are as expected • to use known valid, boundary and erroneous values	2

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

Question	Answer	Marks

3(a)	```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```
	
	One mark for each of the following:
	
	1 Both initialisations
	2 Loop 100 times
	3 Adding individual element to TotalValue **in a loop**
	4 Check if element value is zero **in a loop**
	5 If so increment ZeroCount **in a loop**
	6 Average is calculated after the loop
	7 Both OUTPUT statements, including message and variables
		7

3(b)	<u>PROCEDURE ScanArray</u> <u>(BYREF AverageValue: REAL, _</u> <u>BYREF ZeroCount: INTEGER,</u> <u>ArrayName : ARRAY)</u> One mark for each underlined part Names unimportant but first two parameters must be BYREF	4

Question	Answer	Marks
4(a)(i)	(see table below)	**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"

One mark for each area as outlined. If no row marks then mark by column for columns 2, 3 and 4

Question	Answer	Marks
4(a)(ii)	To remove repeated space characters	**1**
4(a)(iii)	• All leading spaces / spaces at the beginning • will be removed (from the input string)	**2**

Question	Answer	Marks

4(b)	```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```
	
	One mark for each of the following:
	
	1 Array declaration
	2 Additional local variable
	3 Nested loops
	4 Array element assignment **within the inner loop**
	
	```RowIndex``` and ```ColIndex``` can be interchangeable
		4

4(c)	Adaptive Maintenance	**1**

Question	Answer	Marks
5(a)	• Saves development time / no need to write it / can't write it… • Pre-compiled and tested / Increased reliability / reduces chance of error • Is available to all programs	**3**
5(b)	(code below)	**8**

```
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 < 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
```

1 mark for each of the following:

1   Declaration of array of 50 elements
2   Loop to initialise array
3   Conditional loop stopping when all numbers generated
4   Generate a random integer in the range 1 to 50 **in a loop**
5   Count each call to RND() **in a loop**
6   check if the number has already been generated **in a loop**
7   if true, record as generated **in a loop**
8   Output a message plus the Attempts **outside a loop**

Question	Answer	Marks
6(a)	One mark for each of: 1 To make a more manageable / understandable solution 2 Subroutine may be (independently) tested and debugged 3 Program is easier to maintain	**3**

6(b)	'Pseudocode' solution included here for development and clarification of mark scheme.
	Programming language example solutions appear in the Appendix.
	
	<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>
	One mark for each of the following:
	
	1 Function header and close (where appropriate), including parameters
	2 Declaration of local `STRING` variables for CDArtist and CDTitle
	3 `OPEN` and `CLOSE` file for reading (Allow `MyCDs` or `MyCDs.txt`)
	4 `(WHILE)` loop checking for `EOF`
	5 read three lines from file **in a loop**
	6 compare search values with file values **in a loop…**
	7 …If true, set `Location` and exit loop **in a loop**
	8 Return `Location`
		8

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

```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;
```