Cambridge International AS & A Level

Cambridge International Examinations
Cambridge International Advanced Subsidiary and Advanced Level

COMPUTER SCIENCE **9608/22**

Paper 2 Written Paper **May/June 2017**

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 will not enter into discussions about these mark schemes.

Cambridge is publishing the mark schemes for the May/June 2017 series for most Cambridge IGCSE®, Cambridge International A and AS Level and Cambridge Pre-U components, and some Cambridge O Level components.

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>1(a)</td><td>
<table>
<tr><th>Item</th><th>Statement</th><th>Input</th><th>Process</th><th>Output</th></tr>
<tr><td>1</td><td>SomeChars = "Hello World"</td><td></td><td></td><td></td></tr>
<tr><td>2</td><td>OUTPUT RIGHT(String1,5)</td><td></td><td></td><td></td></tr>
<tr><td>3</td><td>READFILE (MyFile, String2)</td><td></td><td></td><td></td></tr>
<tr><td>4</td><td>WRITEFILE (MyFile, "Data is " & String2)</td><td></td><td></td><td></td></tr>
</table>

Mark as follows:

Row 1 as shown
Row 2 no marks if tick in Input column, otherwise 1 mark per tick
Row 3 as shown
Row 4 no marks if tick in Input column, otherwise 1 mark per tick</td><td>6</td></tr>
<tr><td>1(b)(i)</td><td>• Integer / Real / Single / Double / Floating Point / Float
• Boolean</td><td>2</td></tr>
<tr><td>1(b)(ii)</td><td>
<table>
<tr><th>Expression</th><th>Evaluates to</th></tr>
<tr><td>(FlagA AND FlagB) OR FlagC</td><td>TRUE</td></tr>
<tr><td>FlagA AND (FlagB OR FlagC)</td><td>TRUE</td></tr>
<tr><td>(NOT FlagA) OR (NOT FlagC)</td><td>FALSE</td></tr>
</table>

1 mark per answer</td><td>3</td></tr>
<tr><td>1(c)</td><td>
<pre>MyCount ← 101

REPEAT
   OUTPUT MyCount
   MyCount ← MyCount + 2
UNTIL MyCount > 199</pre>
1 mark for each of the following:

• Counter initialisation
• Repeat .. Until loop
• Method for choosing (correct range of) odd numbers
• Output all odd numbers in the range

Note: Counter variable name must be consistent</td><td>4</td></tr>
</table>

Question	Answer	Marks
2(a)	• to increase the level of detail of an algorithm / design... // breaking down a problem / module / task into smaller parts. • .from which the task may be programmed 1 mark per underlined phrase or equivalent	**2**
2(b)	1 mark for first 3 data types – String 1 mark for last data type – Boolean 1 mark for each description: `FileUserID` Stores (User) ID from file `FilePreferredName` Stores (preferred) name from file `IDFoundFlag` True if (User) ID found in file // False if (User) ID not found in file // If `SearchUserID` matches `FileUserID`	**5**
2(c)	1. LOOP through the file until EOF(). 2. OR `SearchUserId` is found 3. READ text line from `UserNames.txt` file **in a loop** 4. EXTRACT `FileUserID` **in a loop** 5. IF `SearchUserId` matches `FileUserID` THEN **in a loop** 6. SET `FilePreferredName` to the name from the file 7. Check if User ID found **not in a loop** 8. OUTPUT appropriate message for both conditions 1 mark per functional equivalent of each numbered statement.	**Max 8**

Question	Answer	Marks

3	`FUNCTION ExCamel (InString: STRING) RETURNS STRING`
	`  DECLARE NextChar : CHAR`
	`  DECLARE OutString : STRING`
	`  DECLARE n : INTEGER`
	`  OutString ← ""  // initialise the return string`
	`  // loop through InString to produce OutString`
	`  FOR n ← 1 TO LENGTH(InString)  // from first to last`
	`  NextChar ← MID(InString, n, 1)  // get next character`
	`  IF NextChar >= 'A' AND NextChar <= 'Z' // check if upper case`
	`  // NextChar = UCASE(NextChar)`
	`  THEN`
	`  IF n > 1  // if not first character`
	`  THEN`
	`  OutString ← OutString & " "  // add space to OutString`
	`  ENDIF`
	`  NextChar ← LCASE(NextChar) // make NextChar lower case`
	`  ENDIF`
	`  OutString ← OutString & NextChar  // add Nextchar to OutString`
	`  ENDFOR`
	`  RETURN OutString  // return value`
	`ENDFUNCTION`
	
	1 mark per underlined word / expression
		Max 11

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>4(a)</td><td>• Functions
• Procedures
• Global / Local variables

1 mark per item</td><td>Max 2</td></tr>
<tr><td>4(b)</td><td>
<table>
<tr><th>Name of parameter passing method</th><th>Value output</th><th>Explanation</th></tr>
<tr><td>(Call) by reference</td><td>5</td><td>• The address of the variable is passed.
• Original value is changed when parameter changed in called module.</td></tr>
<tr><td>(Call) by value</td><td>4</td><td>• A copy of the variable itself is passed.
• Original value not changed when parameter changed in called module.</td></tr>
</table>
Mark as follows:
• 1 mark for each name and value
• 1 mark per bullet in explanation</td><td>6</td></tr>
</table>

Question	Answer	Marks
5(a)(i)	• Any character except colon, space or any alpha-numeric • Reason: character is not in the login information strings	**2**
5(a)(ii)	`DECLARE LogArray : ARRAY[1 : 20] OF STRING` 1 mark per underline	**2**

Question	Answer	Marks

5(b)	Pseudocode solution included here for development and clarification of mark scheme. Programming language example solutions appear in the **Appendix**.
	
	`PROCEDURE LogEvents()`
	`  DECLARE FileData : STRING`
	`  DECLARE ArrayIndex : INTEGER`
	`  OPENFILE "LoginFile.txt" FOR APPEND`
	`  FOR ArrayIndex ← 1 TO 20 //`
	`  IF LogArray[ArrayIndex]<> "****"`
	`  THEN`
	`  FileData ← LogArray[ArrayIndex]`
	`  WRITEFILE ("LoginFile.txt", FileData)`
	`  ENDIF`
	`  ENDFOR`
	`  CLOSEFILE("LoginFile.txt")`
	`ENDPROCEDURE`
	
	1 mark for each of the following:
	
	1. Procedure heading and ending
	2. Declare `ArrayIndex` as integer // commented in python
	3. Open file '`LoginFile`' for append
	4. Correct loop
	5. extract data from array **in a loop**
	6. check for unused element **in a loop**
	7. write data to file **in a loop**
	8. Close the file **outside the loop**
		8

Question	Answer	Marks
6(a)	Pseudocode solution included here for development and clarification of mark scheme. Programming language example solutions appear in the Appendix.	**Max 9**

```
FUNCTION ValidateRegistration(Registration : STRING) RETURNS
                                                        BOOLEAN
  DECLARE UCaseChar, NumChar : INTEGER
  DECLARE NextChar : CHAR
  DECLARE ReturnFlag : BOOLEAN
  DECLARE n : INTEGER

  ReturnFlag ← TRUE
  ValidateRegistration ← True


  IF LEN(Registration) < 6 OR LEN(Registration) > 9  //check
                                                       length
    THEN
      ReturnFlag ← False
    ELSE

      FOR n ← 1 TO 3            //check for 3 upper case alpha
        NextChar ← MID(Registration, n, 1)
        IF NextChar < 'A' AND NextChar > 'Z'
          THEN
            ReturnFlag ← False
        ENDIF
      ENDFOR

      FOR n ← 4 TO 5            //check for 2 numeric
        NextChar ← MID(Registration, n, 1)
        IF NextChar < '0' AND NextChar > '9
          THEN
            ReturnFlag ← False
        ENDIF
      ENDFOR

      FOR n ← 6 TO LEN(Registration) //check remaining
                                                    characters
        NextChar ← MID(Registration, n, 1)
        IF NextChar < 'A' AND NextChar > 'Z'
          THEN
            ReturnFlag ← False
        ENDIF
      ENDFOR
  ENDIF
  RETURN (ReturnFlag)
ENDFUNCTION
```

Question	Answer	Marks
6(a)	1 mark for each of the following: 1. Correct Function heading and ending 2. Check for correct length 3. Extract first three characters 4. Check first three characters are capitals 5. Extract characters four and five 6. Check characters four and five are numeric 7. Extract remaining characters 8. Check remaining characters are capitals 9. Combine all four tests results into a single Boolean value 10. Return a Boolean value	
6(b)	**String1:** (for example, "ABC12XYZ") One mark for a valid string having: • Correct length (between 6 and 9 characters) • 3 capital letters followed by. • 2 numeric characters followed by. • between 1 and 4 capital letters **String2 to String5:** 1 mark for each string **and** explanation (testing different rules of the function) Test strings breaking **one different** rules: • Incorrect length • With incorrect number of capital letters at the start • With non-numeric characters in positions 4 and 5 • With incorrect number of capital letters at the end • Containing an invalid character (not alpha-numeric)	**5**

**** End of Mark Scheme ****

Programming Code Example Solutions

Q5 (b): Visual Basic

```
Sub LogEvents()
    Dim FileData As String
    Dim ArrayIndex As Integer
    FileOpen(1, "LoginFile.txt", OpenMode.Append)
    For ArrayIndex = 1 To 20
        If LogArray(ArrayIndex) <> "****" Then
            FileData = LogArray(ArrayIndex)
            PrintLine(1, FileData)
        End If
    Next
    FileClose(1)
End Sub
```

Alternative:

```
Sub LogEvents()
    Dim FileData As String
    Dim ArrayIndex As Integer
    Dim MyFile As New System.IO.StreamWriter("LoginFile.txt", True)
    For ArrayIndex = 1 To 20
        If LogArray(ArrayIndex) <> "****" Then
            FileData = LogArray(ArrayIndex)
            MyFile.WriteLine(FileData)
        End If
    Next
    MyFile.Close()
End Sub
```

Alternative:

```
Sub LogEvents()
    Dim FileData As String
    Dim ArrayIndex As Integer
    Open "LoginFile.txt" For Append As #1
    For ArrayIndex = 1 To 20
        If LogArray(ArrayIndex) <> "****" Then
            FileData = LogArray(ArrayIndex)
            Print #1, FileData
        End If
    Next
    Close #1
End Sub
```

Q5 (b): Pascal

```
procedure LogEvents;
var FileData : String;
    ArrayIndex : Integer;
    MyFile : Text;
    FileName : String;
begin
    FileName := 'Loginfile.txt';
    AssignFile(MyFile, 'LoginFile.txt');
    if FileExists(FileName) then
        Append(MyFile)
    else
        Rewrite(MyFile);
    for ArrayIndex := 1 to 20 do
    begin
        if LogArray[ArrayIndex] <> '****' then
        begin
            FileData := LogArray[ArrayIndex];
            Writeln(MyFile, FileData);
        end;
    end;
    close(MyFile)
end;
```

Q5 (b): Python

```
# FileData : String
# ArrayIndex : Integer
def LogEvents() :
    MyFile = open("LoginFile.txt", "a")
    for ArrayIndex in range(0, 20) :
        if LogArray[ArrayIndex] != "****" :
            FileData = LogArray[ArrayIndex]
            MyFile.write(FileData + "\n")
    MyFile.close()
```

Alternative:

```
def LogEvents() :
    MyFile = open("LoginFile.txt", "a")
    For FileData in ArrayIndex :
        if FileData!= "****" :
            MyFile.write(FileData + "\n")
    MyFile.close()
```

Q6 (a): Visual Basic

```
Function ValidateRegistration(ByVal Registration As String) As Boolean
    Dim NextChar As Char
    Dim n As Integer
    Dim ReturnFlag As Boolean
    ReturnFlag = True
    If Len(Registration) < 6 Or Len(Registration) > 9 Then
        ReturnFlag = False
    else
        For n = 0 To 2
            NextChar = Registration(n)
            If NextChar < "A" Or NextChar > "Z" Then
                ReturnFlag = False
            End If
        Next
        For n = 3 To 4
            NextChar = Registration(n)
            If NextChar < "0" Or NextChar > "9" Then
                ReturnFlag = False
            End If
        Next
        For n = 5 To Len(Registration) - 1
            NextChar = Registration(n)
            If NextChar < "A" Or NextChar > "Z" Then
                ReturnFlag = False
            End If
        Next
    End If
    ValidateRegistration = ReturnFlag
End Function
```

Alternatives:

```
NextChar = Registration(n)
NextChar = Registration.Chars(n)
```

Q6 (a): Visual Basic

Alternative:

```
Function ValidateRegistration(ByVal Registration As String) As Boolean
    Dim NextChar As String
    Dim n As Integer
    Dim ReturnFlag As Boolean
    ReturnFlag = True
    If Len(Registration) < 6 Or Len(Registration) > 9 Then
        ReturnFlag = False
    else
        For n = 1 To 3
            NextChar = Mid(Registration, n, 1)
            If NextChar < "A" Or NextChar > "Z" Then
                ReturnFlag = False
            End If
        Next
        For n = 4 To 5
            NextChar = Mid(Registration, n, 1)
            If NextChar < "0" Or NextChar > "9" Then
                ReturnFlag = False
            End If
        Next
        For n = 6 To Len(Registration)
            NextChar = Mid(Registration, n, 1)
            If NextChar < "A" Or NextChar > "Z" Then
                ReturnFlag = False
            End If
        Next
    End If
    ValidateRegistration = ReturnFlag
End Function
```

Q6 (a): Pascal

```
function ValidateRegistration(Registration : string) : boolean;
var NextChar : char;
    n : integer;
    ReturnFlag : boolean;
begin
    ReturnFlag := true;
    if ((Length(Registration) < 6) or (Length(Registration) > 9)) then
        ReturnFlag := false
    else
        for n := 1 to 3 do
        begin
            NextChar := Registration[n];
            if ((NextChar < 'A') or (NextChar > 'Z')) then
                ReturnFlag := false;
        end;
        for n := 4 to 5 do
        begin
            NextChar := Registration[n];
            if ((NextChar < '0') or (NextChar > '9')) then
                ReturnFlag := false;
        end;
        for n := 6 to Length(Registration) do
        begin
            NextChar := Registration[n];
            if ((NextChar < 'A') or (NextChar > 'Z')) then
                ReturnFlag := false;
        end;
    ValidateRegistration := ReturnFlag;
end;
```

Alternatives:

```
NextChar := Registration[n];
NextChar := Copy(Registration, n, 1);
```

Q6 (a): Python

```
# Registration : String
# ReturnFlag : boolean
# NextChar : Character
# n : integer
def ValidateRegistration(Registration) :
    ReturnFlag = True
    if len(Registration) < 6 or len(Registration) > 9 :
        ReturnFlag = False
    else :
        for n in range(3) :
            NextChar = Registration[n]
            if NextChar < 'A' or NextChar > 'Z' :
                ReturnFlag = False
        for n in range(3, 5) :
            NextChar = Registration[n]
            if NextChar < '0' or NextChar > '9' :
                ReturnFlag = False
        for n in range(5, len(Registration)) :
            NextChar = Registration[n]
            if NextChar < 'A' or NextChar > 'Z' :
                ReturnFlag = False
    return ReturnFlag
```