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.

This document consists of **8** printed pages.

<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		**Max 11**

```
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

<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><b>Max 2</b></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 <b>and</b> value
- 1 mark per bullet in explanation</td><td><b>6</b></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**

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>5(b)</td><td>

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**

</td><td>**8**</td></tr>
</table>

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**