Cambridge International AS & A Level

Cambridge International Examinations
Cambridge International Advanced Subsidiary and Advanced Level

COMPUTER SCIENCE **9608/23**

Paper 2 **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>Input:
• Enter data into the system // get / receive / read data
• <code>INPUT MyVar // READFILE MyFile, MyString</code>

Process:
• Manipulate / change data in some way // perform a calculation / find a result
• <code>MyChar ← 'X' // MyNum ← MyNum + 1</code>

Output:
• Send data out from the system // display / print / transmit / show data
• <code>OUTPUT "Hello World" // WRITEFILE MyFile, MyString</code>

Mark as follows:
1 mark for each type (in bold)
1 mark for each description and pseudocode example</td><td>7</td></tr>
<tr><td>1(b)(i)</td><td>Boolean</td><td>1</td></tr>
<tr><td>1(b)(ii)</td><td>Logical / Boolean</td><td>1</td></tr>
<tr><td>1(b)(iii)</td><td><table><tr><th>Expression</th><th>Evaluates to</th></tr><tr><td><code>FlagA AND (FlagB OR FlagC)</code></td><td>TRUE</td></tr><tr><td><code>FlagA AND (FlagB AND FlagC)</code></td><td>FALSE</td></tr><tr><td><code>(NOT FlagA) OR (NOT FlagC)</code></td><td>FALSE</td></tr></table>
1 mark per answer</td><td>3</td></tr>
<tr><td>1(c)</td><td><pre>MyCount ← 100

WHILE MyCount < 201

   Output MyCount

   MyCount ← MyCount + 2

ENDWHILE</pre>1 mark for each of the following:

• Counter initialisation
• While ... End loop
• Method for choosing (correct range of) even numbers
• Output all even numbers in the range

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

Question	Answer	Marks
2(a)	Stepwise refinement	**1**
2(b)	1 mark for first 2 data types – String 1 mark for last 2 data types – Boolean 1 mark for each description: `PasswordInput` Stores password entered `UserIDFound` True if user ID found in the file `PasswordValid` True if password entered matches password from file //Input password matches stored password	**5**
2(c)	1. LOOP through the file until EOF… 2. …OR `UserIdInput` is found 3. READ text line from `Password.txt` file **in a loop** 4. SPLIT into `UserID` and password **in a loop** 5. IF `UserIdInput` matches `UserID` from file THEN **in a loop** 6. SET `UserIDFound` to `TRUE` **in a loop** 7. IF `UserIDFound = TRUE` AND `PasswordInput` matches value from file THEN 8. Set `PasswordValid` to `TRUE` Mark as follows: 1 mark per functional equivalent of each numbered statement.	**8**

Question	Answer	Marks

3	`FUNCTION StringClean(Instring STRING) RETURNS STRING`
	`  DECLARE NextChar : CHAR`
	`  DECLARE OutString : STRING`
	`  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 and`
	`  NextChar ← LCASE(NextChar) //convert to lower case`
	
	`  IF NextChar >= 'a' AND NextChar <= 'z' //check if alphabetic`
	`  THEN`
	`  OutString ← OutString & NextChar //add to OutString`
	`  ENDIF`
	`  ENDFOR`
	
	`  RETURN OutString  // return value`
	`ENDFUNCTION`
	
	One mark per **underlined** word / expression
		11

Question	Answer	Marks
4(a)	• The hierarchy of modules • Parameters that are passed between modules // The interface between the modules / • The sequence • Iteration / selection One mark per item	**3**
4(b)	`FUNCTION CardPayment (ParamA : REAL, ParamB : STRING) RETURNS BOOLEAN` One mark per underlined part Order not significant for `ParamA` and `ParamB` Function name and parameter names not important but must be present	**3**

Question	Answer	Marks

5	Pseudocode solution included here for development and clarification of mark scheme.
	Programming language **example** solutions appear in the **Appendix**.
	
	`PROCEDURE SearchFile()`
	
	`DECLARE FileData : STRING`
	`DECLARE MyArrayRow : INTEGER`
	`DECLARE SearchID : STRING`
	`MyArrayRow ← 0 / 1`
	`OPEN "Loginfile.txt" FOR READ`
	`INPUT SearchID`
	
	`WHILE NOT EOF("Loginfile.txt")`
	`  READFILE "Loginfile.txt", Filedata`
	`  IF SearchID = LEFT(FileData,5)`
	`  THEN`
	`  LoginEvents[MyArrayRow,1] ← MID(Filedata, 6, 4)`
	`  LoginEvents[MyArrayRow,2] ← RIGHT(Filedata, 14)`
	`  MyArrayRow ← MyArrayRow + 1`
	`  ENDIF`
	`ENDWHILE`
	
	`CLOSEFILE("LoginFile.txt")`
	
	`ENDPROCEDURE`
	
	1 mark for each of the following:
	
	1. Procedure heading and ending
	2. Declare `MyArrayRow` as integer // commented in python
	3. Initialising `MyArrayRow`
	4. Input `SearchID`
	5. Open file `"LoginFile.txt"` for input / read
	6. Correct loop incorporating `EOF()`
	7. Read a line from the file **in a loop**
	8. Compare `SearchID` with correct data from file **in a loop**
	9. Assign both values to `LoginEvents[MyArray]` **in a loop**
	10. Increment `MyArrayRow` correctly **in a loop**
	11. Close the file **not in a loop**
		10

Question	Answer	Marks
6(a)	Pseudocode solution included here for development and clarification of mark scheme. Programming language solutions appear in the Appendix.	**10**

```
FUNCTION ValidatePassword(InString : STRING) RETURNS
BOOLEAN
DECLARE LCaseChar, UCaseChar, NumChar, n : INTEGER
DECLARE NextChar : CHAR
DECLARE ReturnFlag : BOOLEAN
ReturnFlag ← TRUE
LCaseChar ← 0, UCaseChar ← 0, NumChar ← 0

FOR n ← 1 TO LENGTH(InString)
   NextChar ← MID(InString,n,1)
   IF NextChar > = 'a' AND NextChar < = 'z'
      THEN
         LCaseChar ← LCaseChar + 1
      ELSE
         IF NextChar > = 'A' AND NextChar < = 'Z'
            THEN
               UCaseChar ← UCaseChar + 1
            ELSE
               IF NextChar > = '0' AND NextChar < = '9'
                  THEN
                     NumChar ← NumChar + 1
                  ELSE
                     ReturnFlag ← False    //invalid
                                             character
               ENDIF
         ENDIF
   ENDIF
ENDFOR

IF Not (LCaseChar>=2 AND UCaseChar>= 2 AND NumChar>= 3)
   THEN
      ReturnFlag ← FALSE
ENDIF
RETURN (ReturnFlag)
ENDFUNCTION
```

1 mark for each of the following:

1. Correct Function heading and ending
2. Declaring three counter variables (upper, lower, numeric)
3. Initialising counters
4. Correct loop
5. Picking up `NextChar` from `InString`
6. Check and count number of lower case
7. Check and count number of upper case

Question	Answer	Marks
6(a)	8. Check and count number of numeric 9. Check for invalid character 10. Combine all four tests into a single Boolean value 11. Returning correct Boolean value	

6(b)(i)	**String1: (e.g. "AAbb123")**
	
	One mark for a valid string having:
	• at least 2 uppercase alphabetic
	• at least 2 lowercase alphabetic
	• at least 3 numeric characters
	• No other character
	
	String2 – String5:
	
	One mark for correct string **and** explanation (testing different rules of the function)
	
	Test strings breaking **different** rules:
	• With incorrect numbers of:
	  • Lower case characters
	  • Upper case characters
	  • Numeric characters
	• Containing an invalid character
		5

6(b)(ii)	White Box	**1**
6(b)(iii)	• Testing may be carried out before the modules are developed // not ready for full testing • Module stubs contain simple code to provide a known response // temporary replacement for a called module	**2**

Programming Solutions

Programming Code Example Solutions

Q5 : Visual Basic

```
Sub SearchFile()
   Dim FileData As String
   Dim SearchID As String
   Dim ArrayIndex As Integer

   ArrayIndex = 1
   FileOpen(1, "LoginFile.txt", OpenMode.Input)
   SearchID = Console.Readline()

   Do While Not EOF(1)
      FileData = LineInput(1)
      If SearchID = LEFT(FileData, 5) Then
         LoginEvents(ArrayIndex, 1) = Mid(Filedata, 6, 4)
         LoginEvents(ArrayIndex, 2) = Right(Filedata, 14)
         ArrayIndex = ArrayIndex + 1
      End If
   Loop
   FileClose(1)
End Sub
```

Alternative:

```
Sub SearchFile()
   Dim FileData As String
   Dim SearchID As String
   Dim ArrayIndex As Integer
   Dim MyFile As System.IO.StreamReader

   ArrayIndex = 1
   MyFile = Mycomputer.FileSystem.OpenTextFileReader("Loginfile.txt")
   SearchID = Console.Readline()

   Do While MyFile.Peek < > -1
      FileData = MyFile.Readline()
      If SearchID = LEFT(FileData, 5) Then
         LoginEvents(ArrayIndex, 1) = Mid(Filedata, 6, 4)
         LoginEvents(ArrayIndex, 2) = Right(Filedata, 14)
         ArrayIndex = ArrayIndex + 1
      End If
   Loop
   MyFile.Close
End Sub
```

Q5 : Pascal

```
Procedure SearchFile();

   var FileData   : String;
   var SearchID   : String;
   var ArrayRow   : Integer;
   Var MyFile     : Text;

   Begin
      ArrayRow := 1;
      Assign(MyFile, "Loginfile.txt");
      Reset(MyFile);
      Readln(SearchID);

      While NOT EOF(MyFile) do
      Begin
          Readln(MyFile, FileData)
          IF SearchID = LeftStr(FileData,5) then
              Begin
                  LoginEvents[ArrayRow,1] = Copy(FileData,6,4);
                  LoginEvents[ArrayRow,2] = Rightstr(FileData,14);
                  ArrayRow = ArrayRow + 1
              End;
      End;

      Close(MyFile);
   End.
```

Q5 : Python

```
def SearchFile():

   # FileData : STRING
   # ArrayRow : INTEGER
   # SearchID : STRING

   ArrayRow = 0
   MyFile = open("Loginfile.txt", 'r')
   SearchID = input()
   FileData = MyFile.readline()

   While FileData != ""
      If SearchID = = FileData[:5]                  #First 5 characters
         LoginEvents[ArrayRow][1] = FileData[5:9]   #next 4 characters
         LoginEvents[ArrayRow][2] = FileData[-14:]  #last 14 characters
         ArrayRow = ArrayRow + 1
      FileData = MyFile.readline()


   myFile.close()
   return()
```

Alternative:

```
def SearchFile():

   # FileData : STRING
   # ArrayRow : INTEGER
   # SearchID : STRING

   ArrayRow = 0
   Myfile = open("Loginfile.txt", 'r')
   SearchID = input()
   For FileData in MyFile
      IF SearchID = = FileData[:5]                  #First 5 characters
         LoginEvents[ArrayRow][1] = FileData[5:9]   #next 4 characters
         LoginEvents[ArrayRow][2] = FileData[-14:]  #last 14 characters
         ArrayRow = ArrayRow + 1

   MyFile.close()
   return()
```

Q6 (a): Visual Basic

```
Function ValidatePassword(InString As String) As Boolean
   Dim LCaseChar, UCaseChar, NumChar As Integer
   Dim NextChar As Char
   Dim ReturnFlag As Boolean
   Dim n As Integer
   ReturnFlag = TRUE
   LCaseChar = 0
   UCaseChar = 0
   NumChar = 0

   For n = 1 to Len(InString)
      NextChar = Mid(InString, n, 1)
      If NextChar > = 'a' And NextChar < = 'z' Then
         LCaseChar = LCaseChar + 1
      Else
         If NextChar > = 'A' And NextChar < = 'Z' Then
            UCaseChar = UCaseChar + 1
         Else
            If NextChar > = '0' And NextChar < = '9' Then
               NumChar = NumChar + 1
            Else
               ReturnFlag = False    //invalid character
            End If
         End If
      End If
   Next

   If NOT (LCaseChar > = 2 And UCaseChar > = 2 And NumChar > = 3)Then
      ReturnFlag = FALSE
   End If

   Return(ReturnFlag)

End Function
```

Q6 (a): Pascal

```
Function ValidatePassword(InString : String): Boolean;
   Var LCaseChar, UCaseChar, NumChar : Integer;
   Var NextChar : Char;
   Var ReturnFlag : Boolean;
   Var n : Integer;

   begin
   ReturnFlag := TRUE;
   LCaseChar := 0;
   UCaseChar := 0;
   NumChar := 0;

   For n := 1 to Length(InString) do
      begin
         NextChar := Copy(InString,n,1);
         If NextChar > = 'a' And NextChar < = 'z' Then
            LCaseChar := LCaseChar + 1
         Else If NextChar > = 'A' AND NextChar < = 'Z' Then
                 UCaseChar := UCaseChar + 1
              Else If NextChar > = '0' AND NextChar < = '9' Then
                      NumChar := NumChar + 1
                   Else
                      ReturnFlag := False    //invalid character
      end

   If NOT(LCaseChar > = 2 And UCaseChar > = 2 And NumChar > =3)then
      ReturnFlag := False;


   ValidatePassword := ReturnFlag
   end;
```

Q6 (a): Python

```
def ValidatePassword(InString):
   # lCaseChar, uCaseChar, numChar : INTEGER
   # nextChar : CHAR
   # returnFlag : BOOLEAN
   # n : INTEGER
   returnFlag = TRUE
   lCaseChar = 0
   uCaseChar = 0
   numChar = 0

   for n in range (0, Len(InString))
      nextChar = InString[n]
      If nextChar > = 'a' and nextChar < = 'z':
         lCaseChar = lCaseChar + 1
      ELSE:
         IF nextChar > = 'A' and nextChar < = 'Z':
            uCaseChar = uCaseChar + 1
         ELSE:
            IF nextChar > = '0' and nextChar < = '9':
               numChar = numChar + 1
            ELSE:
               returnFlag = False    //invalid character

   IF Not (lCaseChar > = 2 and uCaseChar > = 2 and numChar > = 3):
      returnFlag = FALSE

   Return (returnFlag)

#next code block
```