
COMPUTER SCIENCE

9608/21

Paper 2 Written Paper

October/November 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 International will not enter into discussions about these mark schemes.

Cambridge International is publishing the mark schemes for the October/November 2017 series for most Cambridge IGCSE[®], Cambridge International A and AS Level components and some Cambridge O Level components.

© IGCSE is a registered trademark.

This document consists of **12** printed pages.

Question	Answer	Marks														
1(a)(i)	<table border="1" data-bbox="392 280 1171 696"> <thead> <tr> <th data-bbox="392 280 603 331">Data value</th> <th data-bbox="603 280 1171 331">Data type</th> </tr> </thead> <tbody> <tr> <td data-bbox="392 331 603 392">27</td> <td data-bbox="603 331 1171 392">INTEGER</td> </tr> <tr> <td data-bbox="392 392 603 452">"27"</td> <td data-bbox="603 392 1171 452">STRING</td> </tr> <tr> <td data-bbox="392 452 603 512">"27.3"</td> <td data-bbox="603 452 1171 512">STRING</td> </tr> <tr> <td data-bbox="392 512 603 573">TRUE</td> <td data-bbox="603 512 1171 573">BOOLEAN</td> </tr> <tr> <td data-bbox="392 573 603 633">27/3/2015</td> <td data-bbox="603 573 1171 633">DATE // DATETIME</td> </tr> <tr> <td data-bbox="392 633 603 696">27.3</td> <td data-bbox="603 633 1171 696">REAL</td> </tr> </tbody> </table> <p data-bbox="244 730 751 801">One mark for each data type Mark first data type given in each case</p>	Data value	Data type	27	INTEGER	"27"	STRING	"27.3"	STRING	TRUE	BOOLEAN	27/3/2015	DATE // DATETIME	27.3	REAL	6
Data value	Data type															
27	INTEGER															
"27"	STRING															
"27.3"	STRING															
TRUE	BOOLEAN															
27/3/2015	DATE // DATETIME															
27.3	REAL															
1(a)(ii)	1D Array // 1DList	2														
1(a)(iii)	<ul data-bbox="244 898 1091 969" style="list-style-type: none"> • Each character is represented by an <u>unique</u> / <u>corresponding</u> • binary code / integer / value 	2														
1(b)	<ul data-bbox="244 1010 1067 1182" style="list-style-type: none"> • When a section of code would be repeated • When a piece of code is needed to perform a specific task • To support modular programming / step wise refinement • Easier to debug / maintain • Built-in / library routines are tried and tested <p data-bbox="244 1218 533 1249">One mark per answer</p>	Max 2														
1(c)	<pre data-bbox="244 1285 724 1473">CASE OF MyVar 1: CALL Proc1() 2: CALL Proc2() 3: CALL Proc3() OTHERWISE OUTPUT "Error" ENDCASE</pre> <p data-bbox="244 1509 429 1541">One mark for:</p> <ul data-bbox="244 1581 639 1727" style="list-style-type: none"> • First line and ENDCASE • All clauses for 1, 2 and 3 • 'OTHERWISE' clause • OUTPUT statement 	4														

Question	Answer	Marks
1(d)	<p>Ability to recognise:</p> <ul style="list-style-type: none"> • selection statement • iteration statement • assignment statements • data declarations / structures / data types / use of variables or objects • modular structure / functions / procedures / subroutines • subroutine parameters • Specific types of statement, e.g. Input, Output, File operations • Code format • Operators <p>Mark as follows: Any two from above, or valid alternative Accept by example</p>	Max 2

Question	Answer	Marks																																										
2(a)	<table border="1"> <thead> <tr> <th>StartNumber</th> <th>EndNumber</th> <th>Divisor</th> <th>NumberFound</th> <th>Number</th> <th>Remainder</th> <th>Output</th> </tr> </thead> <tbody> <tr> <td>11</td> <td>13</td> <td>2</td> <td>0</td> <td>11</td> <td>1</td> <td></td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> <td>12</td> <td>0</td> <td>12</td> </tr> <tr> <td></td> <td></td> <td></td> <td>1</td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> <td>13</td> <td>1</td> <td></td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>Count: 1</td> </tr> </tbody> </table> <p>One mark for correct Number column One mark for correct Remainder column One mark for correct Output</p>	StartNumber	EndNumber	Divisor	NumberFound	Number	Remainder	Output	11	13	2	0	11	1						12	0	12				1								13	1								Count: 1	3
StartNumber	EndNumber	Divisor	NumberFound	Number	Remainder	Output																																						
11	13	2	0	11	1																																							
				12	0	12																																						
			1																																									
				13	1																																							
						Count: 1																																						
2(b)	<p>Mark as follows:</p> <ul style="list-style-type: none"> • For a (given) range of values • Counts the number of times one number (numerator) is an exact divisor of the other • Outputs each numerator (only) • Outputs the count <p>Accept by example No mark for:</p> <ul style="list-style-type: none"> • ...calculate the remainder • ...add one to NumberFound 	3																																										

Question	Answer	Marks
2(c)	<pre> graph TD Start([START]) --> Input[/INPUT StartNumber, EndNumber, Divisor/] Input --> Init1[NumberFound ← 0] Init1 --> Init2[Number ← StartNumber] Init2 --> LoopStart(()) LoopStart --> CalcRem[Remainder ← MODULUS(Number, Divisor)] CalcRem --> DivZero{Remainder = 0?} DivZero -- YES --> OutNum[/OUTPUT Number/] OutNum --> IncFound[Increment NumberFound] DivZero -- NO --> LoopStart IncFound --> LoopStart DivZero --> LoopEnd{Number = EndNumber?} LoopEnd -- NO --> IncNum[Increment Number] IncNum --> LoopStart LoopEnd -- YES --> OutCount[/OUTPUT "Count: " & NumberFound/] OutCount --> Stop([STOP]) </pre>	10

Question	Answer	Marks
2(c)	<p>Mark as follows:</p> <ul style="list-style-type: none">• One mark for START and STOP / END• One mark for bracketed pair• One mark for each of other labelled boxes (shape must be correct for decision box) <p>Decision box outputs must have two outputs and at least one label (Yes / No) Different statement categories should not appear in the same symbol (e.g. assignment and I/O)</p> <p>No mark for symbol (or pair) if parent missing or logically incorrect (except for START/END)</p> <p>Full marks should be awarded for functionally equivalent solutions.</p>	

Question	Answer	Marks
3(a)	<pre> PROCEDURE BubbleSort DECLARE Temp : STRING DECLARE FirstID, SecondID : INTEGER DECLARE NoSwaps : BOOLEAN DECLARE Boundary : INTEGER Declare J : INTEGER Boundary ← 99 REPEAT NoSwaps ← TRUE FOR J ← 1 TO Boundary FirstID ← UserNameArray[J] SecondID ← UserNameArray[J + 1] IF FirstID > SecondID THEN Temp ← UserNameArray[J] UserNameArray[J] ← UserNameArray[J + 1] UserNameArray[J + 1] ← Temp NoSwaps ← FALSE ENDFIF ENDFOR Boundary ← Boundary - 1 UNTIL NoSwaps = TRUE ENDPROCEDURE </pre> <p>Mark as follows:</p> <ol style="list-style-type: none"> 1. Procedure heading and ending (allow array as input parameter) 2. Variable declaration for counter / index (integer) or temp (string) 3. Outer working loop 4. Inner loop with suitable range 5. Correct comparison in a loop 6. Correct swap of complete array element in a loop 7. Set flag to indicate swap in inner loop and resetting in outer loop 8. Reducing Boundary in a loop 	8

Question	Answer	Marks
3(b)	<p>Pseudocode solution included here for development and clarification of mark scheme. Programming language example solutions appear in the Appendix.</p> <pre> PROCEDURE FindRepeats DECLARE i, RepeatCount: INTEGER DECLARE FirstID, SecondID: STRING RepeatCount ← 0 FOR i ← 2 TO 100 FirstID ← LEFT(UsernameArray[i - 1], 6) SecondID ← LEFT(UsernameArray[i], 6) IF FirstID = SecondID THEN RepeatCount ← RepeatCount + 1 OUTPUT(UsernameArray[i]) ENDIF ENDFOR IF RepeatCount = 0 THEN OUTPUT "The array contains no repeated UserIDs" ELSE OUTPUT "There are " & RepeatCount & " repeated userIDs" ENDIF ENDPROCEDURE </pre> <p>Mark as follows (all must be correct syntax for chosen language):</p> <ol style="list-style-type: none"> 1. Procedure heading and ending 2. Variable declaration for INTEGER (comment in Python) and initialisation for RepeatCount (or equivalent name) 3. Loop 4. Extraction of UserID in a loop 5. Correct comparison of consecutive elements... in a loop 6. ...output correct array element (NOT original, only duplicates) in a loop 7. increment RepeatCount following a comparison in a loop 8. Correct conditional statement checking RepeatCount (or equivalent) and then two correct final OUTPUT statements 	Max 8

Question	Answer	Marks
3(c)(i)	<ul style="list-style-type: none"> • Problem definition • Design • Coding / programming • Testing • Documentation • Implementation • Maintenance 	3
3(c)(ii)	<u>Integrated Development Environment</u> or a suitable description	1
3(c)(iii)	<p>Examples include:</p> <ul style="list-style-type: none"> • context sensitive prompts • (dynamic) syntax checking • use of colours to highlight key words / pretty printing • Formatting • Single-stepping • Breakpoints • Report / watch window • (UML) modelling • Compiler/interpreter • Text editor 	Max 2
3(c)(iv)	Run-time	1

Question	Answer	Marks												
4(a)	<table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>Value</th> <th>Formatted String</th> </tr> </thead> <tbody> <tr> <td>1327.5</td> <td>"□ 1327.50"</td> </tr> <tr> <td>1234</td> <td>"□ 1234.00"</td> </tr> <tr> <td>7.456</td> <td>"□□□ 07.45"</td> </tr> </tbody> </table> <p>Leading spaces must be present</p>	Value	Formatted String	1327.5	"□ 1327.50"	1234	"□ 1234.00"	7.456	"□□□ 07.45"	2				
Value	Formatted String													
1327.5	"□ 1327.50"													
1234	"□ 1234.00"													
7.456	"□□□ 07.45"													
4(b)	<table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>Value</th> <th>Required output</th> <th>Mask</th> </tr> </thead> <tbody> <tr> <td>1234.00</td> <td>"1,234.00"</td> <td>"0,000.00"</td> </tr> <tr> <td>3445.66</td> <td>"£3,445.66"</td> <td>"£0,000.00"</td> </tr> <tr> <td>10345.56</td> <td>"\$□□10,345"</td> <td>"\$##00,000"</td> </tr> </tbody> </table> <p>Currency and 'punctuation' symbols must be as shown</p>	Value	Required output	Mask	1234.00	"1,234.00"	"0,000.00"	3445.66	"£3,445.66"	"£0,000.00"	10345.56	"\$□□10,345"	"\$##00,000"	3
Value	Required output	Mask												
1234.00	"1,234.00"	"0,000.00"												
3445.66	"£3,445.66"	"£0,000.00"												
10345.56	"\$□□10,345"	"\$##00,000"												

Question	Answer	Marks
5(a)	<pre>PROCEDURE MakeNewfile DECLARE OldFileLine : STRING DECLARE NewFileLine : STRING OPENFILE "EmailDetails" FOR READ OPENFILE "NewEmailDetails" FOR WRITE WHILE NOT EOF("EmailDetails") READFILE "EmailDetails", OldFileLine NewFileLine ← "00" & OldFileLine WRITEFILE "NewEmailDetails", NewFileLine ENDWHILE CLOSEFILE "EmailDetails" CLOSEFILE "NewEmailDetails" ENDPROCEDURE</pre> <p>Mark as follows:</p> <ol style="list-style-type: none"> 1. Variable declaration of <code>STRING</code> for <code>OldFileLine</code> (or equivalent) 2. Open <code>EmailDetails</code> for <code>READ</code> 3. Open <code>NewEmailDetails</code> for <code>WRITE</code> 4. Correct loop checking for <code>EOF</code> (<code>EmailDetails</code>) 5. Reading a line from <code>EmailDetails</code> in a loop 6. Correct concatenation in a loop 7. Writing a line to <code>NewEmailDetails</code> in a loop <p>Closing both files</p>	8
5(b)	<p>Invalid string examples:</p> <p>A string with nothing before '@'</p> <p>A string with nothing after '@'</p> <p>A string with 1 or 2 characters after '@'</p> <p>A string with no '@' symbol</p> <p>A string with more than one '@' symbol</p> <p>Explanation</p> <p>Sensible explanation mapping each given string to an individual rule</p> <p>One mark for string</p> <p>One mark for explanation</p> <p>Each rule should be tested once only</p>	6

Programming Example Solutions**Q3(b): Visual Basic**

```
Sub FindRepeats()  
    Dim Repeats As Integer  
    Dim i As Integer  
    Dim FirstID As String  
    Dim SecondID As String  
  
    Repeats = 0  
    For i = 1 To 99  
        FirstID = Left(UsernameArray(i), 6)  
        SecondID = Left(UsernameArray(i + 1), 6)  
        If FirstID = SecondID Then  
            Console.WriteLine(UsernameArray(i + 1))  
            Repeats = Repeats + 1  
        End If  
    Next i  
  
    If Repeats = 0 Then  
        Console.WriteLine("The array contains no repeated UserIDs")  
    Else  
        Console.WriteLine("There are " & Repeats & " repeated UserIDs")  
    End If  
  
End Sub
```

Alternative:

```
Sub FindRepeats ()  
  
    Dim RepeatCount, i As Integer  
    Dim FirstID, SecondID As String  
  
    RepeatCount = 0  
    For i = 1 to 99  
        FirstID = Left(UsernameArray(i-1),6)  
        SecondID = Left(UsernameArray(i),6)  
        If FirstID = SecondID then  
            Console.WriteLine (UsernameArray(i))  
            RepeatCount = RepeatCount + 1  
        End If  
    Next i  
  
    If RepeatCount = 0 then  
        Console.WriteLine ("The array contains no repeated UserIDs")  
    Else  
        Console.WriteLine ("There are "& RepeatCount & " repeated UserIDs")  
    End If  
End Sub
```

Q3(b): Pascal

```
procedure FindRepeats ();  
  
var  
  RepeatCount, i : integer;  
  FirstID, SecondID : string;  
  
begin  
  RepeatCount := 0;  
  for i := 1 to 99 do  
    begin  
      FirstID := Copy(UsernameArray[i-1],1,6);  
      SecondID := Copy(UsernameArray[i],1,6);  
      if FirstID = SecondID then  
        begin  
          writeln (UsernameArray[i]);  
          RepeatCount := RepeatCount + 1;  
        end;  
      end;  
  
      if RepeatCount = 0 then  
        writeln ('The array contains no repeated UserIDs')  
      else  
        writeln ('There are ', RepeatCount, ' repeated UserIDs')  
    end;  
end;
```

Q3(b): Python

```
def FindRepeats():
    #Repeats, i Integer
    #FirstID, SecondID string
    Repeats = 0
    for i in range(0, len(UsernameArray)-1):
        FirstID = (UsernameArray[i][:6])
        SecondID = (UsernameArray[i+1][:6])
        if FirstID == SecondID:
            print(UsernameArray[i+1])
            Repeats = Repeats + 1
    if Repeats == 0:
        print("The array contains no repeated UserIDs")
    else:
        print("There are ", Repeats, " repeated UserIDs")
```

Alternative:

```
def FindRepeats ():

    RepeatCount = 0                                ## Defined as an integer

    for i in range (1,100):                        ## depending on next two
lines(0,99) (2,101)
        FirstID = UsernameArray[i-1]              ## Defined as string
        SecondID = UsernameArray[i]               ## Defined as string
        if FirstID[0:6] == SecondID[0:6]:        ## Using split
            print (UsernameArray[i])
            RepeatCount += 1

    if repeatCount == 0:
        print ('The array contains no repeated UserIDs')
    else:
        print ('There are ', RepeatCount, ' repeated UserIDs')
```