

---

**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<sup>®</sup>, Cambridge International A and AS Level components and some Cambridge O Level components.

bestexamhelp.com

---

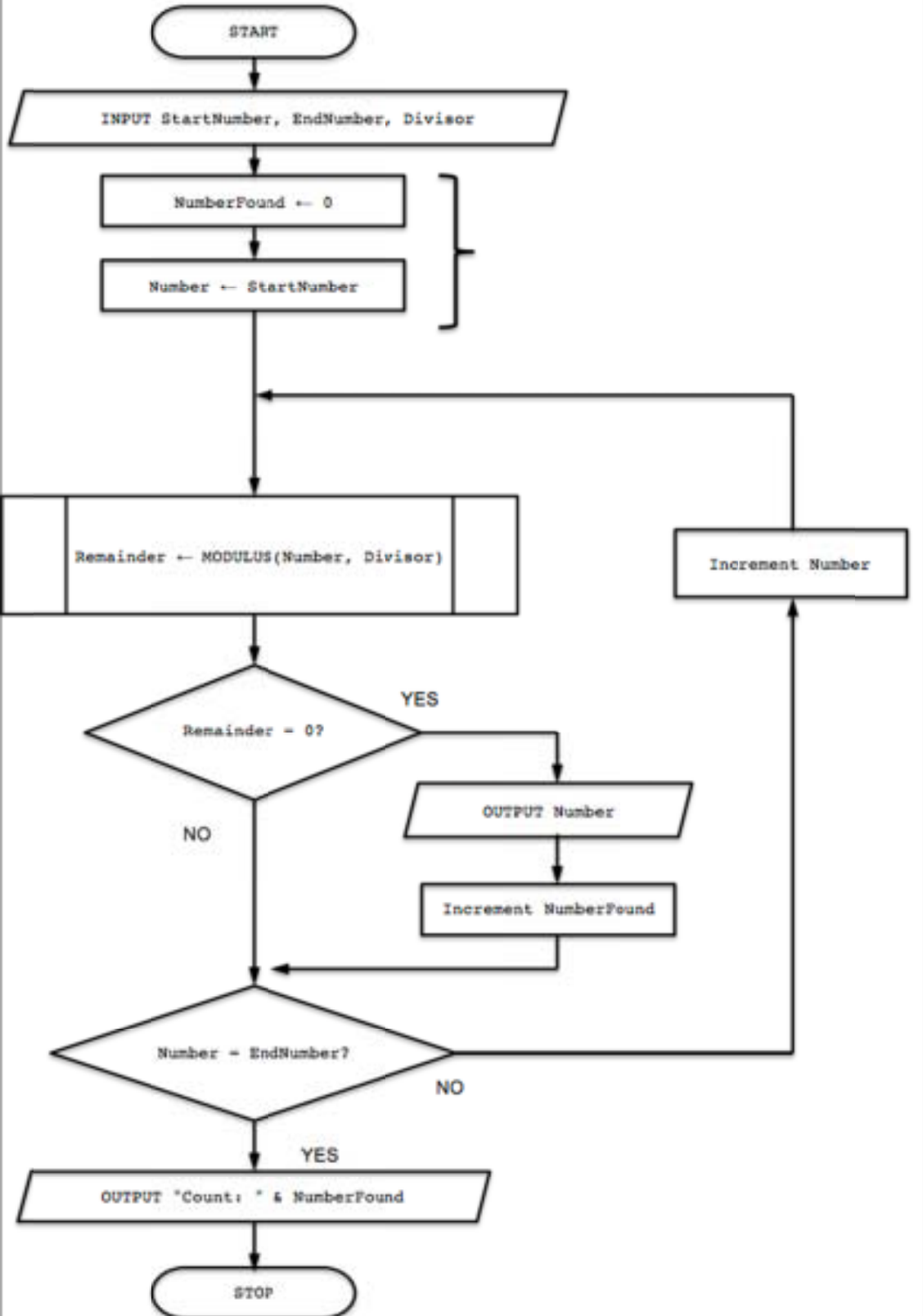
© IGCSE is a registered trademark.

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

Question	Answer	Marks														
1(a)(i)	<table><tr><th>Data value</th><th>Data type</th></tr><tr><td>27</td><td>INTEGER</td></tr><tr><td>"27"</td><td>STRING</td></tr><tr><td>"27.3"</td><td>STRING</td></tr><tr><td>TRUE</td><td>BOOLEAN</td></tr><tr><td>27/3/2015</td><td>DATE // DATETIME</td></tr><tr><td>27.3</td><td>REAL</td></tr></table> <p>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 style="list-style-type: none"><li>Each character is represented by an <u>unique</u> / <u>corresponding</u></li><li>binary code / integer / value</li></ul>	2														
1(b)	<ul style="list-style-type: none"><li>When a section of code would be repeated</li><li>When a piece of code is needed to perform a specific task</li><li>To support modular programming / step wise refinement</li><li>Easier to debug / maintain</li><li>Built-in / library routines are tried and tested</li></ul> <p>One mark per answer</p>	Max 2														
1(c)	<pre>CASE OF MyVar   1: CALL Proc1()   2: CALL Proc2()   3: CALL Proc3()   OTHERWISE OUTPUT "Error" ENDCASE</pre> <p>One mark for:</p> <ul style="list-style-type: none"><li>First line and ENDCASE</li><li>All clauses for 1, 2 and 3</li><li>'OTHERWISE' clause</li><li>OUTPUT statement</li></ul>	4														

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

Question	Answer	Marks																																										
2(a)	<table><tr><th>StartNumber</th><th>EndNumber</th><th>Divisor</th><th>NumberFound</th><th>Number</th><th>Remainder</th><th>Output</th></tr><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></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"><li>• For a (given) range of values</li><li>• Counts the number of times one number (numerator) is an exact divisor of the other</li><li>• Outputs each numerator (only)</li><li>• Outputs the count</li></ul> <p>Accept by example No mark for:</p> <ul style="list-style-type: none"><li>• ...calculate the remainder</li><li>• ...add one to NumberFound</li></ul>	3																																										

Question	Answer	Marks
2(c)	 <pre> graph TD     Start([START]) --&gt; Input[/INPUT StartNumber, EndNumber, Divisor/]     Input --&gt; Init1[NumberFound ← 0]     Init1 --&gt; Init2[Number ← StartNumber]     Init2 --&gt; LoopStart(( ))     LoopStart --&gt; Mod[Remainder ← MODULUS(Number, Divisor)]     Mod --&gt; Div0{Remainder = 0?}     Div0 -- YES --&gt; OutNum[/OUTPUT Number/]     OutNum --&gt; IncFound[Increment NumberFound]     IncFound --&gt; LoopStart     Div0 -- NO --&gt; LoopStart     LoopStart --&gt; EndNum{Number = EndNumber?}     EndNum -- YES --&gt; OutCount[/OUTPUT "Count: " &amp; NumberFound/]     OutCount --&gt; Stop([STOP])     EndNum -- NO --&gt; IncNum[Increment Number]     IncNum --&gt; LoopStart </pre> <p>The flowchart describes a process to count the number of integers between StartNumber and EndNumber (inclusive) that are divisible by Divisor. It begins with a START terminal, followed by an input block for StartNumber, EndNumber, and Divisor. Two initialization steps are shown: NumberFound ← 0 and Number ← StartNumber, which are grouped by a bracket. The process then enters a loop. The first step in the loop is to calculate the remainder of Number divided by Divisor using the MODULUS function. A decision diamond checks if the remainder is 0. If YES, the current Number is output, and NumberFound is incremented. If NO, the process skips the output and increment steps. Both paths lead to a second decision diamond: Number = EndNumber?. If YES, the final count (NumberFound) is output, and the process stops. If NO, the Number is incremented, and the loop repeats from the MODULUS step.</p>	10

Question	Answer	Marks
2(c)	<p>Mark as follows:</p> <ul style="list-style-type: none"><li>• One mark for <b>START and STOP / END</b></li><li>• One mark for bracketed pair</li><li>• One mark for each of other labelled boxes (shape must be correct for decision box)</li></ul> <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 &gt; SecondID         THEN           Temp ← UserNameArray[J]           UserNameArray[J] ← UserNameArray[J + 1]           UserNameArray[J + 1] ← Temp           NoSwaps ← FALSE         ENDIF       ENDFOR       Boundary ← Boundary - 1     UNTIL NoSwaps = TRUE   ENDPROCEDURE </pre> <p>Mark as follows:</p> <ol style="list-style-type: none"> <li>1. Procedure heading and ending (allow array as input parameter)</li> <li>2. Variable declaration for counter / index (integer) <b>or</b> temp (string)</li> <li>3. Outer working loop</li> <li>4. Inner loop with suitable range</li> <li>5. Correct comparison <b>in a loop</b></li> <li>6. Correct swap of complete array element <b>in a loop</b></li> <li>7. Set flag to indicate swap <b>in inner loop and</b> resetting <b>in outer loop</b></li> <li>8. Reducing Boundary <b>in a loop</b></li> </ol>	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 " &amp; RepeatCount &amp; " repeated userIDs"     ENDIF   ENDPROCEDURE </pre> <p>Mark as follows (all must be correct syntax for chosen language):</p> <ol style="list-style-type: none"> <li>1. Procedure heading and ending</li> <li>2. Variable declaration for INTEGER (comment in Python) <b>and</b> initialisation for RepeatCount (or equivalent name)</li> <li>3. Loop</li> <li>4. Extraction of UserID <b>in a loop</b></li> <li>5. Correct comparison of consecutive elements... <b>in a loop</b></li> <li>6. ...output correct array element (NOT original, only duplicates) <b>in a loop</b></li> <li>7. increment RepeatCount following a comparison <b>in a loop</b></li> <li>8. Correct conditional statement checking RepeatCount (or equivalent) and then ... ... two correct final OUTPUT statements</li> </ol>	<b>Max 8</b>

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

Question	Answer	Marks												
4(a)	<table><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><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" &amp; OldFileLine     WRITEFILE "NewEmailDetails", NewFileLine   ENDWHILE    CLOSEFILE "EmailDetails"   CLOSEFILE "NewEmailDetails"  ENDPROCEDURE </pre> <p>Mark as follows:</p> <ol style="list-style-type: none"> <li>1. Variable declaration of <code>STRING</code> for <code>OldFileLine</code> (or equivalent)</li> <li>2. Open <code>EmailDetails</code> for <code>READ</code></li> <li>3. Open <code>NewEmailDetails</code> for <code>WRITE</code></li> <li>4. Correct loop checking for <code>EOF(EmailDetails)</code></li> <li>5. Reading a line from <code>EmailDetails</code> <b>in a loop</b></li> <li>6. Correct concatenation <b>in a loop</b></li> <li>7. Writing a line to <code>NewEmailDetails</code> <b>in a loop</b></li> </ol> <p>Closing both files</p>	<b>8</b>
5(b)	<p><b>Invalid string examples:</b></p> <p>A string with nothing <b>before</b> '@'</p> <p>A string with nothing <b>after</b> '@'</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><b>Explanation</b></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>	<b>6</b>

**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;  
        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')
```