
COMPUTER SCIENCE

9608/22

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="384 280 1166 696"> <thead> <tr> <th data-bbox="384 280 595 331">Data value</th> <th data-bbox="595 280 1166 331">Data type</th> </tr> </thead> <tbody> <tr> <td data-bbox="384 331 595 394">FALSE</td> <td data-bbox="595 331 1166 394">BOOLEAN</td> </tr> <tr> <td data-bbox="384 394 595 456">03/03/2013</td> <td data-bbox="595 394 1166 456">DATE // DATETIME</td> </tr> <tr> <td data-bbox="384 456 595 519">35</td> <td data-bbox="595 456 1166 519">INTEGER</td> </tr> <tr> <td data-bbox="384 519 595 582">"INTEGER"</td> <td data-bbox="595 519 1166 582">STRING</td> </tr> <tr> <td data-bbox="384 582 595 645">3.5</td> <td data-bbox="595 582 1166 645">REAL</td> </tr> <tr> <td data-bbox="384 645 595 696">"35"</td> <td data-bbox="595 645 1166 696">STRING</td> </tr> </tbody> </table> <p data-bbox="236 730 743 797">One mark for each data type Mark first data type given in each case</p> | Data value | Data type | FALSE | BOOLEAN | 03/03/2013 | DATE // DATETIME | 35 | INTEGER | "INTEGER" | STRING | 3.5 | REAL | "35" | STRING | 6 |
| Data value | Data type | | | | | | | | | | | | | | | |
| FALSE | BOOLEAN | | | | | | | | | | | | | | | |
| 03/03/2013 | DATE // DATETIME | | | | | | | | | | | | | | | |
| 35 | INTEGER | | | | | | | | | | | | | | | |
| "INTEGER" | STRING | | | | | | | | | | | | | | | |
| 3.5 | REAL | | | | | | | | | | | | | | | |
| "35" | STRING | | | | | | | | | | | | | | | |
| 1(a)(ii) | 1D Array // 1D List | 1 | | | | | | | | | | | | | | |
| 1(a)(iii) | <p data-bbox="236 898 496 931">Ability to recognise:</p> <ul data-bbox="236 936 1201 1249" 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 data-bbox="236 1290 756 1384">Mark as follows: Any two from above, or valid alternative Accept by example</p> | 2 | | | | | | | | | | | | | | |
| 1(b)(i) | <table border="1" data-bbox="331 1451 679 1832"> <thead> <tr> <th data-bbox="331 1451 679 1503">Data</th> </tr> </thead> <tbody> <tr> <td data-bbox="331 1503 679 1554"></td> </tr> <tr> <td data-bbox="331 1554 679 1606">67 // 0100 0011 // 043h</td> </tr> <tr> <td data-bbox="331 1606 679 1657">65 // 0100 0001 // 041h</td> </tr> <tr> <td data-bbox="331 1657 679 1709">71 // 0100 0111 // 047h</td> </tr> <tr> <td data-bbox="331 1709 679 1760">69 // 0100 0101 // 045h</td> </tr> <tr> <td data-bbox="331 1760 679 1832"></td> </tr> </tbody> </table> <p data-bbox="236 1868 1118 2033">One mark for 67 and 65 One mark for 71 and 69 Accept binary, denary or hex values (hex must be clearly indicated) Max one mark if blank cell anywhere in sequence Ignore any data values before or after the four characters</p> | Data | | 67 // 0100 0011 // 043h | 65 // 0100 0001 // 041h | 71 // 0100 0111 // 047h | 69 // 0100 0101 // 045h | | 2 | | | | | | | |
| Data | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | |
| 67 // 0100 0011 // 043h | | | | | | | | | | | | | | | | |
| 65 // 0100 0001 // 041h | | | | | | | | | | | | | | | | |
| 71 // 0100 0111 // 047h | | | | | | | | | | | | | | | | |
| 69 // 0100 0101 // 045h | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | |

| Question | Answer | Marks |
|----------|---|-------|
| 1(b)(ii) | <ul style="list-style-type: none"> • A value representing the number of characters stored at beginning of string OR • Terminator / special character stored to indicate the end of string <p>One mark for each phrase or equivalent.</p> | 2 |
| 1(c) | <p>Explanation includes:</p> <ul style="list-style-type: none"> • to pass values to/from the subroutine • to produce re-useable code • to avoid global variables • to allow recursion <p>One mark per answer</p> | Max 3 |
| 1(d)(i) | <p>27: MyGrade assigned the value "Fail"</p> <p>101: Output the text "Invalid Value Entered"</p> <p>Ignore minor spelling mistakes</p> | 2 |
| 1(d)(ii) | <pre> IF MyMark >= 75 AND MyMark <=100 THEN MyGrade ← "Distinction" ELSE IF MyMark >= 35 AND MyMark <=74 THEN MyGrade ← "Pass" ELSE IF MyMark >= 0 AND MyMark <=34 THEN MyGrade ← "Fail" ELSE OUTPUT "Invalid value entered" ENDIF ENDIF ENDIF ENDIF </pre> <p>One mark for each of:</p> <ul style="list-style-type: none"> • One correct range test • 'IF' equivalent (nested or not) to three CASE range tests... • ... with three corresponding assignments • Equivalent of CASE OTHERWISE with corresponding OUTPUT statement • Matching (three) ENDIFs (Or one if ELSIFs used) <p>Max 4 if solution doesn't work under all circumstances // is not functionally equivalent to CASE</p> | 5 |

| Question | Answer | Marks |
|----------|--|--------------|
| 2(a) | Mark as follows: <ul style="list-style-type: none"> • To search for a given value in the array • and output the position in the array if the search value is found • if search value not found then output "Not found" | Max 2 |
| 2(b) | <pre> graph TD Start([START]) --> Input[/Input SearchValue/] Input --> FoundFalse[FoundFlag ← FALSE] FoundFalse --> Index1[Index ← 1] Index1 --> LoopStart(()) LoopStart --> Cond1{Index < 101 AND FoundFlag = FALSE?} Cond1 -- YES --> Cond2{ClassName[Index] = SearchVal?} Cond1 -- NO --> FoundFalseCheck{FoundFlag = FALSE?} Cond2 -- YES --> OutputIndex[/OUTPUT Index/] OutputIndex --> FoundTrue[FoundFlag ← TRUE] FoundTrue --> IncIndex[Increment Index] Cond2 -- NO --> IncIndex FoundFalseCheck -- YES --> OutputNotFound[/OUTPUT "Not found"/] FoundFalseCheck -- NO --> IncIndex IncIndex --> LoopStart OutputNotFound --> Stop([STOP]) </pre> | 9 |

| Question | Answer | Marks |
|----------|--|-------|
| 2(b) | Mark as follows: <ul style="list-style-type: none"> • One mark for <code>START</code> and <code>STOP / END</code> • One mark for each bracketed pair • One mark for each of other labelled symbol (decision box shape must be correct) • Allow F/T from incorrect decision symbol Full marks should be awarded for functionally equivalent solutions. | |

| Question | Answer | Marks | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|-------------|---|---|-------|------------|----|------------------------------------|---------------------|----|------------------------|-----------------------------|----|---------------------|--|----|-----------------------|-----------------|----|-------------|------------------------|-------|--------------------------|---|-------|------------------|--|----|-------------------------------|---------------------------|----|-------------------------------------|-------------------------------------|-------|----------------|-----------------------|-------|
| 3 | <table border="1"> <thead> <tr> <th>Line number</th> <th>Error</th> <th>Correction</th> </tr> </thead> <tbody> <tr> <td>01</td> <td>Wrong procedure name – "SortArray"</td> <td>PROCEDURE ArraySort</td> </tr> <tr> <td>02</td> <td>Wrong data type - CHAR</td> <td>DECLARE Temp: STRING</td> </tr> <tr> <td>03</td> <td>Variables undefined</td> <td>DECLARE FirstID, SecondID, I, J : INTEGER</td> </tr> <tr> <td>04</td> <td>Wrong 'Value2' of 100</td> <td>FOR I ← 1 TO 99</td> </tr> <tr> <td>05</td> <td>Wrong range</td> <td>FOR J ← 1 TO (100 - I)</td> </tr> <tr> <td>06/07</td> <td>Wrong function - MODULUS</td> <td>Replace MODULUS with TONUM: FirstID ← TONUM(LEFT(Product[J],</td> </tr> <tr> <td>06/07</td> <td>Wrong value of 6</td> <td>Should be 4: FirstID ← TONUM(LEFT(Product[J],</td> </tr> <tr> <td>10</td> <td>Assigning wrong value to Temp</td> <td>Temp ← Product[J]</td> </tr> <tr> <td>11</td> <td>Assigning wrong value to Product[I]</td> <td>Product[J] ← Product[J + 1]</td> </tr> <tr> <td>13/14</td> <td>Lines reversed</td> <td>13 ENDIF 14 ENDFOR</td> </tr> </tbody> </table> <p>One mark for each correct row</p> | Line number | Error | Correction | 01 | Wrong procedure name – "SortArray" | PROCEDURE ArraySort | 02 | Wrong data type - CHAR | DECLARE Temp: STRING | 03 | Variables undefined | DECLARE FirstID, SecondID, I, J : INTEGER | 04 | Wrong 'Value2' of 100 | FOR I ← 1 TO 99 | 05 | Wrong range | FOR J ← 1 TO (100 - I) | 06/07 | Wrong function - MODULUS | Replace MODULUS with TONUM: FirstID ← TONUM (LEFT(Product[J], | 06/07 | Wrong value of 6 | Should be 4: FirstID ← TONUM(LEFT(Product[J], | 10 | Assigning wrong value to Temp | Temp ← Product [J] | 11 | Assigning wrong value to Product[I] | Product [J] ← Product[J + 1] | 13/14 | Lines reversed | 13 ENDIF 14 ENDFOR | Max 8 |
| Line number | Error | Correction | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 01 | Wrong procedure name – "SortArray" | PROCEDURE ArraySort | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 02 | Wrong data type - CHAR | DECLARE Temp: STRING | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 03 | Variables undefined | DECLARE FirstID, SecondID, I, J : INTEGER | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 04 | Wrong 'Value2' of 100 | FOR I ← 1 TO 99 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 05 | Wrong range | FOR J ← 1 TO (100 - I) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 06/07 | Wrong function - MODULUS | Replace MODULUS with TONUM: FirstID ← TONUM (LEFT(Product[J], | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 06/07 | Wrong value of 6 | Should be 4: FirstID ← TONUM(LEFT(Product[J], | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 10 | Assigning wrong value to Temp | Temp ← Product [J] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 11 | Assigning wrong value to Product[I] | Product [J] ← Product[J + 1] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 13/14 | Lines reversed | 13 ENDIF 14 ENDFOR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

| Question | Answer | Marks |
|----------|--|-------|
| 4(a) | <p>Pseudocode solution included here for development and clarification of mark scheme. Programming language solutions appear in the Appendix.</p> <pre> PROCEDURE TestRandom (Repetitions : INTEGER) DECLARE Frequency : ARRAY [1 : 10] OF INTEGER DECLARE Expected : REAL / INTEGER //allow either DECLARE NextRandom : INTEGER DECLARE N : INTEGER FOR N ← 1 TO 10 Frequency[N] ← 0 ENDFOR Expected ← INT(Repetitions / 10) CALL RANDOMIZE() //Set random seed FOR N ← 1 TO Repetitions NextRandom ← INT(RND() * 10) + 1 Frequency[NextRandom] ← Frequency[NextRandom] + 1 ENDFOR OUTPUT "The expected frequency is " & Expected OUTPUT "Number Frequency Difference" FOR N ← 1 TO 10 OUTPUT N & " " & Frequency[N] & " " & Frequency[N] - Expected ENDFOR ENDPROCEDURE </pre> <p>Mark as follows:</p> <ol style="list-style-type: none"> 1. Procedure heading (including parameter) 2. Array declaration – 10 or 11 elements 3. Array declaration – data type 4. Variable declaration for a loop counter (integer) or expected frequency (integer or real) 5. Variable declaration for next random value <p>(For Python solutions, mark points 1 to 4 may be gained by suitable comments)</p> <ol style="list-style-type: none"> 6. Initialise all elements of array 7. To set all elements to zero 8. Calculate expected frequency | 16 |

| Question | Answer | Marks |
|----------|--|--------------|
| 4(a) | 9. Loop to generate required number of random values 10. Use of relevant <code>RANDOM()</code> function in a loop 11. Generate random integer value in the range 1 to 10 in a loop 12. Increment (array) element in a loop 13. Output expected frequency message not in any loop 14. Output column header text 15. (Loop to) output each row 16. ... including three correct values (spaces optional) | |
| 4(b) | <ul style="list-style-type: none"> • Single-stepping <ul style="list-style-type: none"> – to allow program statements to be executed one at a time • Breakpoints <ul style="list-style-type: none"> – to pause / stop the program at a specific line / statement • Variable / expression watch window <ul style="list-style-type: none"> – to monitor the value of variables / expressions as the program is run <p>One mark for each Feature (text as above or equivalent) + 1 for meaningful explanation of use in context.</p> | 6 |
| 4(c) | <p>Program is probably working correctly if:</p> <ul style="list-style-type: none"> • Header is present giving frequency as 20 • Column headers are present • All rows are present (1 to 10) • The difference is calculated correctly • Output is formatted correctly • Total differences should be zero • Sum of Frequencies should be 200 | Max 2 |

| Question | Answer | Marks |
|----------|---|-------|
| 5 | <pre> PROCEDURE RemoveDetails DECLARE FileLine: STRING DECLARE MemberToDelete: STRING OPENFILE "EmailDetails.txt" FOR READ OPENFILE "NewEmailDetails.txt" FOR WRITE INPUT MembershipNumber WHILE NOT EOF("EmailDetails.txt") READFILE "EmailDetails.txt", FileLine IF LEFT(FileLine, 4) <> MembershipNumber THEN WRITEFILE "NewEmailDetails.txt", FileLine ENDIF ENDWHILE CLOSEFILE "EmailDetails.txt" CLOSEFILE "NewEmailDetails.txt" ENDPROCEDURE </pre> <p>Mark as follows:</p> <ol style="list-style-type: none"> 1. Procedure declaration and end. No parameters. 2. Variable declaration of <code>STRING</code> for variable <code>FileLine</code> (or similar) 3. Input the <code>MembershipNumber</code> of the person who has left 4. Open <code>EmailDetails</code> for <code>READ</code> 5. Open <code>NewEmailDetails</code> for <code>WRITE</code> 6. Correct loop checking for <code>EOF</code> (<code>EmailDetails</code>) 7. Reading a line from <code>EmailDetails.txt</code> in a loop 8. Correct check for <code>MemberToDelete</code> in a loop 9. Writing a line to <code>NewEmailDetails.txt</code> in a loop 10. Closing both files (not in a loop) | Max 9 |

Appendix - Program Code Example Solutions**Q4 (a): Visual Basic**

```

Dim random As New Random()

Sub TestRandom(ByVal repetitions As Integer)
    Dim randinrange As Integer
    Dim i As Integer
    Dim num(1 To 10) As Integer
    Dim freq As Integer
    Dim difference As Integer

    freq = repetitions / 10    'calculate expected frequency

    For i = 1 To 10            'initialise array to store total frequencies
        num(i) = 0
    Next i

    For i = 1 To repetitions 'generate random numbers & increment
appropriate freq
        randinrange = random.Next(1, 11)
        num(randinrange) = num(randinrange) + 1
    Next i

    Console.WriteLine("The expected frequency is " & freq)    'report header
    Console.WriteLine("Number    Frequency    Difference")    'column headers

    For i = 1 To 10    'calc & display difference between expected and actual
freq
        difference = num(i) - freq
        Console.WriteLine(i & "    " & num(i) & "    " & difference)
    Next i

End Sub

```

Other possible ways of calculating a random number in VB include:

```

    randinrange = CInt(Math.Floor((upperbound - lowerbound + 1) * Rnd())) +
lowerbound

    randinrange = math.round((Rnd()*9)+1)

    randinrange = CInt(Math.Ceiling(Rnd() * 9)

```

Q4 (a): Pascal

```
procedure TestRandom(var Repetitions : integer);
  var
    Frequency : array[1..10] : integer;
    Expected, NextRandom, N : integer;

  begin
    Expected := Round(Repetitions/10);
    for N := 1 to 10 do
      Frequency[N] := 0;

    for N := 1 to Repetitions do
      begin
        NextRandom := random(10)+1;
        Frequency[NextRandom] := Frequency[NextRandom]+1;
      end;

    writeln ('The expected frequency is ', Expected);
    writeln ('Number Frequency Difference');

    for N := 1 to 10 do
      writeln ('    ',N,'          ',Frequency[N],'          ',Frequency[N]-
Expected);
    end;
  end;
```

Q4 (a): Python

```
# frequency is an array from 1 to 10 of type integer;
# nextNumber is an integer which stores the created random number
# expected is an integer which stores the expected frequency of each number

def TestRandom (repetitions):
    import random
    frequency = [0 for i in range(1,11)] # initialise each frequency count
    to 0

    expected = repetitions / 10

    for i in range(1, repetitions + 1):
        nextNumber = random.randint(1,10)
        frequency[nextNumber] = frequency[nextNumber]+ 1

    print ("The expected frequency is ", expected)
    print("  Number  Frequency  Difference")

    for i in range(1,11):
        print ("      ", i, "      ", frequency[i], "      ", frequency[i] -
expected)
```

Alternative:

```
def TestRandom (repetitions):
    expected = repetitions / 10 ## initialised as real/integer
                                ## NextRandom and N defined as integers
    frequency =[0,0,0,0,0,0,0,0,0,0,0] ## defined as an array and
initialised to zero

    for n in range (0,repetitions):
        nextNumber = randint(1, 10)
        frequency[nextNumber] += 1

    print ('The expected frequency is ', expected)

    print ('Number  Frequency  Difference')
    for n in range (1, 11):
        print (n, '      ', frequency[n], '      ', frequency[n] - expected)
```

Alternative:

```
frequency =[0]*11 ## alternate way to initialise array to zero
frequency =[] ## empty array/list
```

Alternative:

```
for n in range (1,11):
    frequency[n-1] = 0 ##alternate way to initialise array to zero
```

Alternative:

```
for n in range (0,11): ##alternate way to initialise array to zero
    frequency.append(0)
```

AlevelCSOnline.CF