<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
<th>Marks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1(a)(i)</td>
<td></td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>Data value</td>
<td>Data type</td>
</tr>
<tr>
<td></td>
<td>FALSE</td>
<td>BOOLEAN</td>
</tr>
<tr>
<td></td>
<td>03/03/2013</td>
<td>DATE // DATETIME</td>
</tr>
<tr>
<td></td>
<td>35</td>
<td>INTEGER</td>
</tr>
<tr>
<td></td>
<td>&quot;INTEGER&quot;</td>
<td>STRING</td>
</tr>
<tr>
<td></td>
<td>3.5</td>
<td>REAL</td>
</tr>
<tr>
<td></td>
<td>&quot;35&quot;</td>
<td>STRING</td>
</tr>
</tbody>
</table>

One mark for each data type
Mark first data type given in each case

| 1(a)(ii) | 1D Array // 1D List | 1 |

| 1(a)(iii) | Ability to recognise: |
|           | 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 |

Mark as follows:
Any two from above, or valid alternative
Accept by example

<table>
<thead>
<tr>
<th>1(b)(i)</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>67 // 0100 0011 // 043h</td>
</tr>
<tr>
<td></td>
<td>65 // 0100 0001 // 041h</td>
</tr>
<tr>
<td></td>
<td>71 // 0100 0111 // 047h</td>
</tr>
<tr>
<td></td>
<td>69 // 0100 0101 // 045h</td>
</tr>
</tbody>
</table>

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
<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
<th>Marks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1(b)(ii)</td>
<td>• A value representing the number of characters ... ... stored at beginning of string OR • Terminator / special character ... ... stored to indicate the end of string One mark for each phrase or equivalent.</td>
<td>2</td>
</tr>
<tr>
<td>1(c)</td>
<td>Explanation includes: • to pass values to/from the subroutine • to produce re-useable code • to avoid global variables • to allow recursion One mark per answer</td>
<td>Max 3</td>
</tr>
<tr>
<td>1(d)(i)</td>
<td>27: MyGrade assigned the value &quot;Fail&quot; 101: Output the text &quot;Invalid Value Entered&quot; Ignore minor spelling mistakes</td>
<td>2</td>
</tr>
<tr>
<td>1(d)(ii)</td>
<td>IF MyMark &gt;= 75 AND MyMark &lt;=100 THEN MyGrade ← &quot;Distinction&quot; ELSE IF MyMark &gt;= 35 AND MyMark &lt;=74 THEN MyGrade ← &quot;Pass&quot; ELSE IF MyMark &gt;= 0 AND MyMark &lt;=34 THEN MyGrade ← &quot;Fail&quot; ELSE OUTPUT &quot;Invalid value entered&quot; ENDIF ENDIF ENDIF One mark for each of: • 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) Max 4 if solution doesn’t work under all circumstances // is not functionally equivalent to CASE</td>
<td>5</td>
</tr>
<tr>
<td>Question</td>
<td>Answer</td>
<td>Marks</td>
</tr>
<tr>
<td>----------</td>
<td>--------</td>
<td>-------</td>
</tr>
</tbody>
</table>
| 2(a)     | Mark as follows:  
- 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)     | ![Flowchart](image) | 9 |
### Question 2(b)

Mark as follows:
- One mark for **START** and **STOP** / **END**
- 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 3

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

Pseudocode solution included here for development and clarification of mark scheme. Programming language solutions appear in the Appendix.

```plaintext
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
```

Mark as follows:
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

(For Python solutions, mark points 1 to 4 may be gained by suitable comments)

6. Initialise all elements of array
7. To set all elements to zero
8. Calculate expected frequency

<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
<th>Marks</th>
</tr>
</thead>
<tbody>
<tr>
<td>4(a)</td>
<td>Pseudocode solution included here for development and clarification of mark scheme. Programming language solutions appear in the Appendix.</td>
<td>16</td>
</tr>
<tr>
<td>Question</td>
<td>Answer</td>
<td>Marks</td>
</tr>
<tr>
<td>----------</td>
<td>--------</td>
<td>-------</td>
</tr>
</tbody>
</table>
| 4(a) | 9. Loop to generate required number of random values  
10. Use of relevant \texttt{RANDOM()} function \textbf{in a loop}  
11. Generate random integer value in the range 1 to 10 \textbf{in a loop}  
12. Increment (array) element \textbf{in a loop}  
13. Output expected frequency message \textbf{not in any loop}  
14. Output column header text  
15. (Loop to) output each row  
16. ... including three correct values (spaces optional) | |
| 4(b) | • Single-stepping  
– to allow program statements to be executed one at a time  
• Breakpoints  
– to pause / stop the program at a specific line / statement  
• Variable / expression watch window  
– to monitor the value of variables / expressions as the program is run | 6 |
| 4(c) | Program is probably working correctly if:  
• 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 |
<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
</table>
| 5        | 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  

Mark as follows:  
1. Procedure declaration and end. No parameters.  
2. Variable declaration of STRING for variable FileLine (or similar)  
3. Input the MembershipNumber of the person who has left  
4. Open EmailDetails for READ  
5. Open NewEmailDetails for WRITE  
6. Correct loop checking for EOF(EmailDetails)  
7. Reading a line from EmailDetails.txt \textbf{in a loop}  
8. Correct check for MemberToDelete \textbf{in a loop}  
9. Writing a line to NewEmailDetails.txt \textbf{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;
        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;
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:

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