Introduction:

Algorithms are ways to solve a problem and should be evaluated for speed, cost, resource usage, and so on. Problem situations also often have constraints and algorithms have to work within those. And accordingly, the performance goals of the algorithm might differ too – it has to execute within so much time, or within so much memory. There can be multiple ways to solve a problem and hence evaluation is important to select the ‘optimal’ way given your constraints and / or goals.
The problems below require knowledge of for loop, if-else construct so please review that before distributing the worksheets.

Questions: (* questions can be used for evaluation)

1. Teacher has to group 300 students for a picnic as follows. Girls more than 14 years don’t need bus, 12-14 years and living within 5 kms radius don’t need bus but more than that do. Girls under 12 need the bus, as do boys under 12. Boys 12 years or more do not need the bus. Teacher writes a program to do the classification as below:

```
For each student
If it is a girl
      If she is > 14 years
      Need_bus = false
      End if
 If she is >= 12 and < 14 years
 If she is living within 5 kms
      Need_bus = false
      Else
            Need_bus = true
      End if
      End if
 If she is < 12 years
      Need_bus = true
      End if
      End if
If it is a boy
If he is < 12 years
      Need_bus = true
      End if
If he is >= 12 years
Need_bus = false
End if
End if
End-for-loop
```

Assume that it is costly to execute ‘if’ statements. Can you optimize the algorithm to use fewer ‘ifs’ and also using ‘else’ where appropriate?

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

2. A researcher has to produce data on how often the 3 letters b, m and p occur in English text. He is given hundreds of pieces of long text. He must scan each piece, and keep updating the overall occurrence % of these vowels. He writes this algorithm:

```
For each piece of text
        For each letter in the text
                If letter = b, then countb++
        End-for-loop
        pc_b = (countb / length of text) X 100
        overall_b = [overall_b + pc_b] / 2
        For each letter in the text
                If letter = m, then countm++
        End-for-loop
        pc_m = (countm / length of text) X 100
        overall_m = [overall_m + pc_m] / 2
        For each letter in the text
                If letter = p, then countp++

        End-for-loop
        pc_p = (countp / length of text) X 100
        overall_p = [overall_p + pc_p] / 2
        End-of-for-loop
```

a) Can you evaluate if this algorithm is optimally written and optimize it?
b) As the algorithm executes, can it teach itself which letter to look for first so as to optimize execution?

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

ANSWER SHEET

1. Rewritten algorithm with 4 ifs instead of 8

```
For each student
        If < 12 years old
                Need_bus = true
        Else if boy
                Need_bus = false
        Else
                /** so we reach here if it is a girl >= 12 years
```

```
                    Need_bus = true
                    If she is > 14 years
                        Need_bus = false
                    Else if she is living within 5 kms
                        Need_bus = false
                    End if
                End if
            End-for-loop
```

2. First modification: We are going through a piece of text 3 times – once for each letter. Instead we should go through it only once and check all 3 letters one after the other.

```
For each piece of text
For each letter in the text
    If letter = p, then countp++
    Else if letter = b, then countb++
    Else if letter = m, then countm++
End-for-loop
pc_p = (countp / length of text) X 100; overall_p = [overall_p + pc_p] / 2
pc_b = (countb / length of text) X 100; overall_b = [overall_b + pc_b] / 2
pc_m = (countm / length of text) X 100; overall_m = [overall_m + pc_m] / 2
 End-of-for-loop
```

Second modification:
Inside the loop we are checking first for b, then m, then p. Note that the first 'if' executes for every letter, the second for every letter except the b's, and the third 'if' for every letter except the b's and the m's. So, in English, if p occurs the most often, we are unnecessarily checking b first, then m, and then p! So we create a list of 3 entries that tells us in which order to check the letters. We re-arrange the list after every piece of text we evaluate. We could even do that after every 5 or 10 pieces if we want.

```
/* create a 3-item list. Each item has a letter, a count, a pc and an overall. */

list(1).letter = b; list(1).count = 0; list(1).pc = 0; list(1).overall = 0
list(2).letter = m; list(2).count = 0; list(2).pc = 0; list(2).overall = 0
list(3).letter = p; list(3).count = 0; list(3).pc = 0; list(3).overall = 0
    For each piece of text
        For I = 1 to 3
            list(i).count = 0; list(i).pc = 0
        End-for-loop
        For each letter in the text
            For I = 1 to 3
            If letter = list(i).letter then list(i).count++
        End-for-loop
    End-for-loop
    For I = 1 to 3
        List(i).pc = [list(i).count/length-of-text] X 100
        List(i).overall = [list(i).overall + list(i).pc] / 2
    End-for-loop
    If list[1].overall < list[2].overall then swap-list-item-1-and-2
    If list[1].overall < list[3].overall then swap-list-item-1-and-3
    If list[2].overall < list[3].overall then swap-list-item-2-and-3
End-of-for-loop
```