Errata for
Visual Basic and Algorithmic Thinking for the Complete Beginner
Second Edition

14.6 Review Exercises

3. Write a Visual Basic program that prompts the user to enter his or her name and then creates a secret password consisting of three letters (in lowercase) randomly picked up from his or her name, and a random four-digit number. For example, if the user enters "Vassilis Bouras" a secret password can probably be one of "sar1359" or "vbs7281" or "bor1459". Space characters are not allowed in the secret password.

31.7 How to Add User-Entered Values to a One-Dimensional Array

There is nothing new here. Instead of reading a value from the keyboard and assigning that value to a variable, you can directly assign that value to a specific array element. The next Visual Basic program prompts the user to enter the names of four people, and assigns them to the elements at index positions 0, 1, 2, and 3, of the array `names`.

Exercise 34.1-4 Merging Two-Dimensional Arrays

project_34_1_4

```
Const COLUMNS = 4

Sub Main(args As String())
  Dim i, j As Integer

  Dim a(,) As Integer = {
    {10, 11, 12, 85},
    {3, 1, 5, 10},
    {-1, 2, -5, -10}
  }

  Dim b(,) As Integer = {
    {10, 11, 16, 33},
    {11, 13, 5, 55},
    {-1, -2, -4, 44},
    {55, 33, 77, 12},
    {-110, 120, 132, 43}
  }

  Dim rows_of_a As Integer = a.Length / COLUMNS
  Dim rows_of_b As Integer = b.Length / COLUMNS

  'Create array new_arr
  Dim new_arr(rows_of_a + rows_of_b - 1, COLUMNS - 1) As Integer
  For i = 0 To rows_of_a - 1
    For j = 0 To COLUMNS - 1
      new_arr(i, j) = a(i, j)
    Next
  Next
  For i = 0 To rows_of_b - 1
    For j = 0 To COLUMNS - 1
      new_arr(rows_of_a + i, j) = b(i, j)
```

```
    Next
  Next

  'Display array new_arr
  For i = 0 To rows_of_a + rows_of_b - 1
    For j = 0 To COLUMNS - 1
      Console.Write(new_arr(i, j) & vbTab)
    Next
    Console.WriteLine()
  Next
End Sub
```

Exercise 34.1-5 Creating Two Arrays – Separating Positive from Negative Values

Note that the arrays `pos` *and* `neg` *contain a total number of* `pos_index` *and* `neg_index` *elements respectively. This is why the two last loop control structures iterate until variable* `i` *reaches values* `pos_index - 1` *and* `neg_index - 1`*, respectively, and not until* `ELEMENTS - 1`*, as you may mistakenly expect. ~~Obviously the sum of pos_index + neg_index equals to ELEMENTS.~~*