LAB GOALS

To create and manipulate arrays.
1) Array length, and array upper bound
2) Initialize, Print, Randomize
3) Passing array parameters to procedures

Step 1: Create a new console project named “Console_Application_Array_Manipulation”. Make sure to save your new project in your I101 or B100 folder.

Step 2: Start by copying and pasting the following code in your console application. Then compile and run your program.

```vbnet
Option Explicit On
Option Strict On
Module Module1
    Sub Main()
        Dim TestScores(10) As Integer
        Console.WriteLine("Array length = {0}", TestScores.Length)
        Console.WriteLine("Array upper bound = {0}", TestScores.GetUpperBound(0))

        Console.ReadLine() ' pause the program
    End Sub
End Module
```

Note that the above code will create an array of ten integers (actually 11). Note that the array is an object and therefore has some properties, such as `Length` and methods such as `GetUpperBound()`.

**More about properties and methods later!**

Step 3: Let’s write some code that initializes all the elements of the array to “-1”. To do so we need a FOR loop.

```vbnet
For Index = 0 To 10
    TestScores(Index) = -1
Next Index
```

Note that the above is better expressed as:

```vbnet
For Index = 0 To TestScores.GetUpperBound(0)
    TestScores(Index) = -1
Next Index
```

Run: Compile and Run your program to make sure it does not have any syntax errors. At this point the program is not very exciting since it does not really show you the content of the array. So, let’s fix that.

Step 4: Let’s now write the code that prints all the elements of the array. We will use a FOR loop to do this as well. The basic structure of the FOR loop remains the same. The only thing that changes is what we do inside the FOR loop.

```vbnet
For Index = 0 To TestScores.GetUpperBound(0)
    Console.WriteLine(TestScores(Index))
Next Index
```

Run: Compile and Run your program again. You should be able to see the output as a series of “-1” in the output. How many are there?
Step 5: Let’s do some more operations with this array. The first of these could be to load the array with RANDOM numbers (in the range of 0 and 100). Once again, we will use a FOR loop to do this. The structure of the FOR loop remains the same. The only thing that changes is what we do inside the loop. Type the following in your program.

```vba
Randomize() 'Generate a Random Seed
For Index = 0 To TestScores.GetUpperBound(0)
    TestScores(Index) = CInt(Rnd() * 100)
Next Index
```

After loading the array with RANDOM numbers, we need to also print the array so that we can make sure our code is working.

```vba
For Index = 0 To TestScores.GetUpperBound(0)
    Console.WriteLine(TestScores(Index))
Next Index
```

Run: Compile and Run your program.

**NOTE:** Please note that we have now duplicated the code for printing the array twice within this program. In fact, as we do different operations, we may find ourselves duplicating this code a few more times. Which means that this code is a perfect candidate for a sub program. In fact, let’s try to put this code inside a procedure and simply call the procedure from within the Main() program.

Step 6: Creating a procedure for Printing the Array. Let’s create the specification of the procedure. Basically a shell, in which the code will eventually reside. Note that the integer array must be passed to the procedure, so that it can print its content. Also, note that TheArray is passed ByVal. Passing the array ByVal means that we have no intention to modify its content; we simply are going to print its content.

```vba
Private Sub Print_Array(ByVal TheArray() As Integer)
    PUT YOUR CODE HERE
End Sub
```

Now, copy and past the code for printing the array, inside the procedure. Note that the name of the array as it is passed to the procedure is called TheArray, not TestScore. So, make sure that you adjust the name. Don’t worry,
we are still printing the content of the original array. So, here is the code.

```vbnet
Private Sub Print_Array(ByVal TheArray() As Integer)
    For Index = 0 To TheArray.GetUpperBound(0)
        Console.WriteLine(TheArray(Index))
    Next Index
End Sub
```

Run: Compile and Run your program.

Step 7: Now, we need to remove the duplicate code for printing the array, and simply call the PrintArray() procedure.

Run: Compile and Run your program.
Step 8: **Challenge:** Can you create a procedure named `Initialize_Array()` that will initialize the array elements to “-1”, and simply call that from your main program? Can you do the same for putting random numbers in the array? Make sure to send the array to the `Initialize_Array()` procedure ByRef. This is important, since you are planning to change the content of the array inside the procedure. (We will have more discussion on this topic in class)

Note that if you do the above Challenge correctly, your main program should look like the following:

```vbnet
Sub Main()
    Dim TestScores(10) As Integer
    Console.WriteLine("Array length = {0}", TestScores.Length)
    Console.WriteLine("Array upper bound = {0}", TestScores.GetUpperBound(0))
    Initialize_Array(TestScores)
    Print_Array(TestScores)
    Randomize_Array(TestScores)
    Print_Array(TestScores)
    Console.ReadLine() ' pause the program
End Sub
```