To create user defined functions.  To understand parameter passing.

Throughout this semester, we have been using many built-in functions. In fact Console.ReadLine(), CInt(), CStr(), and CDbl() have been some common examples of built-in function that we have used for weeks.

A function is very similar to a procedure, so everything that we have learned about procedures, apply to functions. However, they are syntactically a little different and they always return at least one value. The typical definition and use of the function is as follows:

**Function Definition:**

```
Private Function FunctionName([Optional Parameters]) As DataType
    [Statements]
    Return(Expression)
End Function
```

**Function Use:**

```
Variable = FunctionName([parameters])
```

In this lab we will extend our knowledge about modularized programming by creating a few user-defined functions. We will also review the concept of parameter passing.

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

Step 2: Create the following shell structure for a console application. Then compile and run it to make sure it works.

```
Option Explicit On
Option Strict On
Module Module1
    Sub Main()
        Console.ReadLine()      ' pause the program
    End Sub
End Module
```

Step 3: Now, let’s create a new user defined function called `IsEven()`. This function will be used to determine if a given integer number is EVEN or not. The function should return a Boolean value “TRUE” if the number is even, and “FALSE” otherwise.

```
Private Function isEven(ByVal theNumber As Integer) As Boolean
    If theNumber Mod 2 = 0 Then
        Return (True)
    Else
        Return (False)
    End If
End Function
```

The above is a new user-defined function. When called upon, this function will check the content of its
“theNumber” parameter to see if it is EVEN (divisible by 2), if so, it will return TRUE to the calling module. The result can then be used in a conditional (IF-THEN-ELSE) or a loop (DO –While, Do- Until), etc.

Run: Compile and Run your program. What is the output? Why aren't you getting any output?

Step 4: To execute the module built in Step 3, we must call it. In order to call this module, type the following lines right before the Console.ReadLine() statement in the main() module.

```csharp
If isEven(5) Then
    Console.WriteLine("5 is even")
Else
    Console.WriteLine("5 is odd")
End If
```

Run: Compile and Run your program.

Step 5: Let’s use the new isEven( ) function within a FOR loop. Place the following FOR loop in the main() sub program.

```csharp
Module Module1
    Sub Main()
        Dim Counter As Integer
        For Counter = 1 To 10
            If isEven(Counter) Then
                Console.WriteLine(Counter)
            End If
        Next
        Console.ReadLine() ' pause the program
    End Sub
End Module
```

Run: Compile and Run your program. What is the result?

Step 5: Let’s use the new isEven( ) function within a WHILE loop. Place the following WHILE loop in the main() sub program.

```csharp
Module Module1
    Sub Main()
        Dim aValue As Integer
        Console.Write("Please enter an integer (Odd number to stop):")
        aValue = CInt(Console.ReadLine()) ' Read an integer from the keyboard
        Do While isEven(aValue)
            Console.WriteLine("{0} is Even.", aValue)
            Console.Write("Please enter an integer (Odd number to stop):")
            aValue = CInt(Console.ReadLine)
        Loop
        Console.WriteLine("{0} is Odd and stops the loop.", aValue)
        Console.ReadLine() ' pause the program
    End Sub
End Module
```

Run: Compile and Run your program. What is the result?
Step 6: Let’s use the new isEven() function within a WHILE loop that checks the loop-condition at the end of the loop. Place the following loop in the main() sub program.

```vbnet
Module Module1

Sub Main()

Dim aValue As Integer

Do
    Console.Write("Please enter an integer (Odd number to stop):")
    aValue = CInt(Console.ReadLine()) ' Read an integer from the keyboard
    If isEven(aValue) = True Then
        Console.WriteLine("{0} is Even.", aValue)
    Else
        Console.WriteLine("{0} is Odd.", aValue)
    End If

Loop While isEven(aValue) = True

Console.WriteLine("{0} is Odd and stops the loop.", aValue)
Console.ReadLine() ' pause the program

End Sub
End Module
```

Run: Compile and Run your program. How is the result different that that obtained from Step5?

On your own:

- Write a function IsOdd(), which determines if the parameter sent to it is ODD. It returns TRUE if the parameter is ODD, and returns FALSE otherwise.

- Write a function called IndianaTax() which takes the “PurchasePrice” as a parameter (DOUBLE). The function should calculate the sales tax and return it to the calling module as a DOUBLE.

- Write a function called IndianaTax() which takes two parameters “PurchasePrice” and “TaxRate” (both as DOUBLE). The function should calculate the sales tax and return it to the calling module as a DOUBLE.