LAB GOALS

Understanding the SDLC.
To develop algorithms in pseudocode. Converting your algorithm to code.

The goal of this lab is to further familiarize you with the art and science of computer programming. As you may recall from earlier lectures, the software development life cycle (SDLC) involves the following steps:
1) Requirement specification
2) Analysis
3) Design
4) Implementation
5) Testing
6) Maintenance

We have already applied the above steps to a number of small problems. Our goal in this lab is to introduce you to another small problem and guide you through the development of a proper algorithmic solution. In other words, we will go through step 1 through 5 of the SDLC. You are further encouraged to take your algorithmic solution and convert them to VB programs.

Problem Definition:

Write a function called IsPrime() that accepts a positive integer as a parameter and returns a Boolean value TRUE if the number received is a prime number. Otherwise, the function should return FALSE.

Step 1: Requirement specification:

Right away, you should ask yourself:
1) Do I understand the problem definition? Is this problem ambiguous? If so, what is the ambiguity?
2) Do I know what a prime number is? What resources do I have that helps me understand the problem? Can I “Google” it? (http://en.wikipedia.org/wiki/Prime_number)
3) Manually check the following numbers to see if they are prime numbers:
   13, 17, 27, 45, 71, 73, 99
   While checking to see if the above numbers are prime, observe your approach? What are you doing to determine if the number is a prime or not?

When you have figured out the answer to the questions posed above go to the next page.
Step 2: Hopefully, by now you have some ideas about how one determines if a number is a prime or not. One simple method of determining the primality of a given number is described below: (taken from Wikipedia.org)

“The property of being prime is called **primality**. Verifying the primality of a given number \( n \) can be done by **trial division**. The simplest trial division method tests whether \( n \) is a multiple of an integer \( m \) between 2 and \( \sqrt{n} \). If \( n \) is a multiple of any of these integers then it is a **composite number**, and so not prime; if it is not a multiple of any of these integers then it is prime. As this method requires up to \( \sqrt{n} \) trial divisions, it is only suitable for relatively small values of \( n \). More sophisticated algorithms, which are much more efficient than trial division, have been devised to test the primality of large numbers.”

http://en.wikipedia.org/wiki/Prime_number

Step 3: If you still are unsure about how to do this, see the example below:

**Problem:**
Is 13 a prime?

**Solution:**
\[ \sqrt{13} = 3.x \]
Is 13 divisible by 2? NO
Is 13 divisible by 3? NO
Is 13 divisible by 4? NO

The answer is YES, is a prime because it does not have any divisors between 2 and \( \sqrt{13} \).

**Problem:**
Is 123 a prime?

**Solution:**
\[ \sqrt{123} = 11.x \]
Is 123 divisible by 2? NO
Is 123 divisible by 3? YES

We don't need to go any further. The answer is NO, 123 is not a prime because it is divisible by 3 (a number between 2 and \( \sqrt{123} \)).

Step 4: Analysis:

Determine the tools you need to solve this problem. (input, output, facts, formulas, constraints, etc. then identify some possible solutions.

When you have analyzed the problem and identified some possible solutions, write your solution in the box below:
Step 5: If you are having difficulty coming up with a solution, see the box below:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1) I need to ask the user to enter a positive integer. (TheNumber)</td>
<td></td>
</tr>
<tr>
<td>2) I need to find out how I can get the square root of that number. Does my programming language have a built-in sqrt() function?</td>
<td></td>
</tr>
<tr>
<td>3) I need some sort of a loop that allows me to check and see if TheNumber is divisible by any number between 2 and sqrt(TheNumber). If so, TheNumber is not a prime. Otherwise, it is.</td>
<td></td>
</tr>
<tr>
<td>4) I need to know how I can tell if a number is divisible by some other number. (Does MOD do the job?)</td>
<td></td>
</tr>
<tr>
<td>5) I also need to make sure TheNumber is not 2. If TheNumber is equal 1 or 2, then I know it is a prime.</td>
<td></td>
</tr>
</tbody>
</table>

Step 6: Design

Write the pseudocode for the above problem. Remember that you may have to revise the pseudocode 2, 3, or more times before it is ready to be implemented. This refinement process is very normal. Use the box below to write your pseudocode.

<table>
<thead>
<tr>
<th>Version 1</th>
<th>Version 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Version 3</th>
<th>Version 4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Hint: VB does have a built-in function for calculating the square root. Check for Math.Sqrt( )
So your solution may seem similar to the one below:

- Print “Please enter a positive integer”
- Read TheNumber
- IsPrime = TRUE  ‘ we’ll assume the number is a prime until proven wrong
- If TheNumber = 1 or TheNumber = 2 then
  - IsPrime = TRUE
- Else
  - TheSQRT = sqrt(TheNumber)
  - TheSQRT = INT(TheSQRT)
  - For Counter = 2 to TheSQRT
    - If (TheNumber MOD Counter) = 0 Then
      - IsPrime = FALSE
    - Next Counter
  - Print IsPrime
  - End if

The next step is to try to implement this solution. Convert the pseudocode above to VB and write it in the box below:
Your implementation should look something like the following:

```vbs
Dim TheNumber, TheSQRT As Integer
Dim IsPrime As Boolean

Console.WriteLine("Enter a positive integer")
TheNumber = CInt(Console.ReadLine())

IsPrime = True ' let's assume the number is already prime until we prove otherwise

If (TheNumber = 1) Or (TheNumber = 2) Then
    IsPrime = True
Else
    TheSQRT = CInt(Math.Sqrt(TheNumber))
    For Counter = 2 To TheSQRT
        If (TheNumber Mod Counter) = 0 Then
            IsPrime = False
        End If
    Next Counter

Console.WriteLine("Primality of {0} is = {1}", TheNumber, IsPrime)
```

Now convert the solution into a Function called IsPrime()

```vbs
Option Explicit On
Option Strict On

Module Module1

Sub Main()
    Dim TheNumber As Integer
    Dim TheResult As Boolean

    For TheNumber = 1 To 20
        TheResult = IsPrime(TheNumber)
        Console.WriteLine("Primality of {0} is = {1}", TheNumber, TheResult)
    Next

    Console.ReadLine()
End Sub

Private Function IsPrime(ByVal TheNumber As Integer) As Boolean
    Dim TheSQRT, Counter As Integer
    IsPrime = True ' let's assume the number is already prime until we prove otherwise

    If (TheNumber = 1) Or (TheNumber = 2) Then
        IsPrime = True
    Else
        TheSQRT = CInt(Math.Sqrt(TheNumber))
        For Counter = 2 To TheSQRT
            If (TheNumber Mod Counter) = 0 Then
                IsPrime = False
            End If
        Next Counter

    Return (IsPrime)
End Function
End Module
```

On your own:

- Design an algorithm calculating the factorial of a number.