Mathematics Integrated with Computers and Applications I

An Undergraduate Programming-Based Mathematics Course, Brock University (Canada)

Two-hour Weekly Labs: Guidelines (Updated in 2017)

Guidelines originally created in 2002 by
Dr. Bill Ralph (Brock University)

Programming Language: *Visual Basic.NET* through *Visual Studio* development environment

Textbook: Visual Basic.Net Step by Step by Michael Halvorson, version 2010

For some information about the MICA courses, see:

Ralph, W. (2001). Mathematics takes an exciting new direction with MICA program. *Brock Teaching,* 1(1), 1.

Buteau, Muller, & Ralph (2015). Integration of Programming in the Undergraduate Mathematics Program at Brock University. In Online Proceedings of the *Math + Coding Symposium,* London (Canada), June 2015.

For a design analysis of the MICA I lab sessions, see:

Buteau, C., & Muller, E. (2014). Teaching roles in a technology intensive core undergraduate mathematics course. In A. Clark-Wilson, O. Robutti, & N. Sinclair (eds.), *The mathematics teacher in the digital era* (pp. 163-185). Springer Netherlands.

Contact (alphabetical order):

Chantal Buteau, Brock University: cbuteau@brocku.ca
Eric Muller, Brock University: emuller@brocku.ca
Bill Ralph, Brock University: bralph@brocku.ca

Overview of programming concepts learned through activities during the two-hour weekly MICA I laboratory sessions

(from Buteau, Muller, & Ralph, 2015)

Week	Programming concepts	Main Lab Activity	Assignment submission
1	Designing a graphical interface	Writing (i.e., copy line by line) a first program	
2	Programming variables	"Hello World" & creating a basic calculator,	
3	Conditional structure, loops	Check primality of an integer	
4	Function and Sub-procedures	Create table of n, n^2, & n^3	EO 1
5	Arrays	Powers in Z_n	
6		gcd, Euler's function, inverse in Z_n	
7	Graphics *(& systemic exploration of a concept)*	Draw different shapes & exploration of Euler's theorem	EO 2
8	Graph of a function	Graphing a parabola, points, and coordinate system	
9		EO for the exploration of the dynamical system of the logistic function (numerical table and cobweb)	
10	*(systemic exploration of a dynamical system)*	Exploration, guided by instructor, of the dynamical system based on the logistic function	EO 3
11		Individual work on final project	
12		Individual work on final project	EO 4 *(in following week)*

Lab #1: Getting Started

In labs, you will be guided as you learn VB.net basic programming for the purpose of investigating mathematics conjectures, concepts, theorems, or real-world situations.

Lab #1 Getting Started (using the 2010 textbook version):

- Complete chapter 2 (intro) - skip 'deploying your application' at p. 62.
- Complete chapter 3 until p. 84
- Read 'A word about technology' section at pp. 89-91
- One or two days *before* your next lab, complete the two after-lab exercises.

After-lab #1 exercises:

1. Open **myHello** project.
   a) Change the text 'Hello, world!' to 'Bonjour, le monde!'
   b) Change the font to a fancier one, and set the font size to 18.
   c) Change the button text 'Ok' to 'Hello, world! in French'.
   d) (optional) repeat a) and c) with another language
   e) Save your changes

2. Create a new project, named **myMe**, with 2 buttons, 1 picture box, and 1 label.
   a) In the label, add the text ‘Hi, my name is ...’ (replace ... with your first name). Use a fancy font and a light yellow background.
   b) In picturebox, add a picture of yourself.
   c) In button1, add the text 'Show “me”', and in button2, add the text 'Exit'

When the program is launched, it shows your name (label1). When clicking on button1, your picture appears. When clicking on button2, the program exits.

Lab #2: Variables

Lab #2 Programming Variables:

- Complete chapter 3 (pp. 84 - 91) from last week, including reading 'A word about technology' section at pp.89-91
- Complete chapter 5

REMINDERS:

Working with the book exercises:

- Go to Computer -> Class Data (P:) -> Visual Basic 2010 Step-by-Step, and copy e.g. Chapter 2 folder, and paste it on your desktop
- Open the files (e.g., LuckySeven.sln) you want to work with from this folder.

Saving your projects:

- In the menu, select 'file', then 'save all'

Saving your project in a different folder:

- Copy the project folder (the whole folder not only the solution file), and paste it where you want to save it.

Lab #3: Decision Structures & Loops

PART I – Prime Project

First create a form with 1) an "End" Button, 2) a "Start" Button for which you write the VB.Net code below for testing the primality of a positive integer, and 3) two TextBoxes, one for the user to enter the number to be tested, and one for displaying the results "Prime" or "Not Prime":

```
Dim i, p As Long            'the long type is for huge integers
Dim IsPrime As Boolean      'Boolean variables can only be true or false
p=TextBox1.Text             'assigns the value the user enters in the Textbox1 to the variable p
TextBox2.Text =""           'clear textbox2
IsPrime = True              'assume p is prime until we get evidence to the contrary

'test all the possible factors up  to square root of p
For i=2 To Int(Math.Sqrt(p))
        If (p Mod i) = 0 Then 'test to see if i divides evenly into p
                IsPrime=False 'if i divides evenly into p then p is not prime so set IsPrime=false
        End If
Next
If IsPrime Or (p=2) Then    'put the conclusion in the TextBox2 and handle the case where p=2
        TextBox2.Text= "Prime"
Else
        TextBox2.Text= "Not Prime"
End If
```

Then work on the following exercises:

1 - How would you modify this code to handle the fact that 1 is not a prime number?

2 - Find a number that will take the program approximately 10 seconds to test

3 - How could you improve the efficiency of the code (i.e. reduce the number of computation steps)?

Optional - Enhance your interface and code to calculate the number of prime numbers smaller than 100.

Optional Challenge 1 - Modify your program to look for the first prime after 20,831,323. ***(Hint:** use an outer loop that runs through the numbers starting at 20,831,324 and an inner loop that tests the primality of the number.)*

Optional Challenge 2 - Write a program in which the user can enter the values a, b, and c and the program gives the solutions of the equation ax^2+bx+c=0. Design a user friendly interface in such a way that the cases where there are no roots, one root or two roots are all handled separately and clearly. Your interface should have clear instructions for the user and it should always be very what is going on.

PART II – Textbook exercises

Complete chapters 6 & 7 (until p.202) – *carefully read Tables 6.1, 6.2, and 6.3*

Lab #4: Procedures (Sub & Functions) & Debugging

Lab #4 consists of:

- Complete exercises A & B
- Complete chapter 10 (all) about *Sub & Function procedures*
- Work on Assignment 1
- *(Optional) complete chapter 8 about* debugging

A. The prime function

Open your program from Lab #3 which tested the primality of a positive integer. Create another button 'Prime Function' which will result in also testing the primality of a given integer. By executing commands differently, it will read the input from the user, conduct computations 'outside' the button_click procedure and return only the result which will be displayed:

- Create a Function procedure that takes an integer as an argument, and returns the primality test result (True or False):

```
Function PrimalityFunction(ByVal n As Integer) As Boolean
    Dim IsPrime As Boolean
    Dim i as Long
    IsPrime = True
    For i=2 To Int(Math.Sqrt(n))
            If (p Mod i) = 0 Then
                IsPrime=False
            Exit For
            End If
    Next
    Return (IsPrime)
End Function
```

- Add the following code under the 'Prime Function' button:

```
Dim p As Long
p=TextBox1.Text
TextBox2.Text =""
    'if p is prime, display 'prime', and else, display 'not prime'
If PrimalityFunction(p) And p<>1 Then
            TextBox2.Text= "Prime"
Else
    TextBox2.Text= "Not Prime"
End If
```

B. Table of Data Sub-procedure

Create a new project that displays squares and cubes of positive integers up to an upper bound, and their sums: the user enters an integer upper bound, and vb.net displays a table with all data, with the sums of squares and of cubes in the last line.

Design your interface (two buttons and two textboxes) and implement the code (in the button_click procedure) according to the following coding design idea:

- Declare a variable named Upperbound of type Integer
- Read the upper bound value from the user
- Call the sub procedure by writing: ShowTable(Upperbound)

Then create a Sub procedure (outside the button_click procedure) that constructs the table as a string and displays it in Textbox2:

```
Sub ShowTable(ByVal n As Integer)
    Dim Output As String
    Dim i, Square, Cube As Integer
    Dim SumOfSquares, SumOfCubes as Long
    For i = 1 to n
        Square = i*i
        Cube = i*Square
        SumOfSquares+=Square
        SumOfCubes+=Cube
        Output=Output & i & vbTab & Square & vbTab & Cube & vbCrlf
        'vbTab inserts a tab; vbCrlf inserts a change line
    Next
    Output  = "Sums " & vbTab & SumsOfSquares & vbTab & SumsOfCubes & vbCrlf
    TextBox2.Text = Output
End Sub
```

** you need to adjust two properties of textbox 2 in order to show the table: multiline set to true and use a vertical scrollbar

NOTES

1) Example of a function

'This function returns the value of the hypothenus of a right-angle triangle, given the length of the two smaller sides

```
Function Hypothenus(ByVal a As Double, ByVal b As Double) As Double
'ByVal is used to declare the arguments
        Dim sum As Double
        sum = a*a + b*b
        Return Math.Sqrt(sum)
End Function
```

in Button:

```
        Dim sideA, sideB, sideC as Double
        sideA=Textbox1.Text
        sideB=Textbox2.Text
        sideC=Hypothenus(sideA,sideB) 'call the function Hypothenus which return a Double value, and
                                       assign the value to variable sideC
```

2) Example of Sub procedure

'This procedure greets you in a label

```
Sub Greetings()
        Label1.Text = "Greeting " & YourName
End Sub
```

At top of the form

```
Dim YourName As String =""
```

in Button:

```
YourName = "Buteau"
Greetings() 'call the Sub procedure called Greetings
```

LAB #5: Powers of Integers in Z_n & Arrays

Lab #5 consists of:

- Complete the Powers of Integers in Z_n exercise
- Complete chapters 11 about arrays
- Complete the after-lab exercises

What you need to complete by hand *BEFORE* starting this lab

- Write the multiplication table of Z_8
- Draw the Z_8-clock, and calculate 2^{13} mod 8 by multiplying 2 by itself 13 times in the clock.
- Similarly with 3^{11} mod 8
- Similarly with 6^8 mod 8

Powers of Integers in Z_n exercise

Write a program that displays all powers of an integer, modulo an integer: the user enters an integer **n**, the modulus **Modulus**, and the maximum power **Maxp**, and vb.net displays a table with all powers.

PART A – powers in Z_n

1. Create a project called "PowersModuloM"
2. Design an interface to display the wished powers modulo Modulus
3. Write the whole program in the procedure of a 'Start' button and that displays, in a textbox, the powers in a table using a string, named "Output" (built inside the loop):
   - Declare four variables, named n, Modulus, Maxp, Power, exponent of type integer
   - Declare a variable named Output of type String
   - Read the values for n, Modulus, and Maxp from the user
   - Assign the value 1 to Power
   - Write a 'for loop' for exponent from 1 to Maxp:

     Power=Power*n Mod Modulus

     Output = … 'complete this statement
   - Display the output (table) in a textbox

PART B – Saving/reading the powers in an array

We will now modify this program so that not only you will display the powers, but you will also record them in an *array* during the loop. The advantage of saving the powers in an array is that once you have calculated and recorded them, you can access them at any time later in the program; a functionality that is not possible when we directly display our results in a string.

Modify your project "PowersModuloM" as follow:

- Declare **powers** as a global variable of type array of integers of size 10000 by typing the following at the top of your form:

```
Dim powers(10000) As Integer
```

- Create another button 'Start - array', and when this button is clicked, it reads the users' **n**, **Modulus,** and **Maxp**

- Write a Sub procedure **ComputePowers** that takes values **n**, **Modulus**, and **Maxp**, and that saves the powers in **powers**, and displays the output table in a textbox by typing the following:

```
Sub ComputePowers(ByVal n As Integer, ByVal Modulus As Integer, ByVal Maxp As Integer)
        Dim power, i As Integer
        Dim Output As String
        Power = 1
        For i=1 to Maxp
                Power=Power*n Mod Modulus
                powers(i)=Power 'this saves the value of Power, i.e., n^i in the ith ' position of the array
                Output = … ' complete this statement
        Next
   'write the command line that displays the table, i.e. Output, in a Textbox
End Sub
```

- Test your ‘Start-array’ button. It should show you the same table as before (but all powers have also been saved in an array)

- Add a third button ‘Powers’ and two textboxes to your interface. The user will enter an exponent, and the program will display the following sentence: “The power of **n** to the **exponent** in Z_**Modulus** is **x**” where the variables in bold should be replaced by values:
  a. Declare two variables, named x and expo, of type integer
  b. Read the value of expo from the user
  c. Assign to **x** the expo*th* value in the array **powers(10000)**
  d. Display “The power of **n** to the **exponent** in Z_**Modulus** is **x**” with values in textbox

Lab #6: RSA Encryption Preparation Lab

Complete Part I (individually) and Part II 1-8 (in groups of 2-4)

Part I: Preparing for RSA encryption

In the same project (i.e., use one interface with possibly many buttons and textboxes; label clearly what each button and textbox is about), complete the following tasks A-C:

***Calculation by hand in preparation for CODE A:** Use Euclid's algorithm to calculate the gcd of i) 56 and 430, and ii) 336 and 240*

CODE A: Write a FRIENDLY program that uses a function called "Gcd" to return the value of gcd(n,m) for two input values of **n** and **m**. Your program should contain the line of code:

```
Function Gcd(ByVal n As Long, ByVal m As Long) As Long
```

You can use the following algorithm (what is returned?):

```
p = maximum value of m and n
q = minimum value of m and n
r = remainder of p divided by q
while r > 0 do
    p takes the value of q
    q takes the value of r
    r is the remainder of p divided by q
return ? ‘ What should it return?
```

***Calculation by hand in preparation for CODE B:** Calculate $\phi(12)$, $\phi(13)$, and $\phi(14)$.*

CODE B: Definition: The Euler "phi" function, $\phi(n)$, is defined to be the number of integers in {1, 2, 3, ..., n-1} that are relatively prime to n.

Write a FRIENDLY program that uses a function called "Euler" to return the value of $\phi(n)$ for an input value of **n**. Calculate by hand $\phi(n)$ for **n**=6,7,8, and 20, and check your answer with your program. You can use the following algorithm:

```
count =0
for i = 1 to n-1
    if gcd(i,n)=1, then count += 1
return ? ‘ What should it return?
```

***Calculation by hand in preparation for CODE C:** Determine whether i) 5 is invertible in Z_{13} or not, and if so, what its inverse is; ii) Similarly for 5 in $Z_{15;}$ iii) Similarly for 5 in Z_{16}*

CODE C: Write a FRIENDLY program that:

i) allows a user to input two numbers **e** and **n**

ii) displays a message saying that e doesn't have an inverse in Zn <u>or</u> displays a message giving the inverse of **e** in Zn, depending on the result.

PART II: Let's do it: RSA Encryption!

Part A: Construct your public key

1) RSA public key encryption uses giant primes of 200 or more digits. For this lab, pick two different primes less than 100 whose product is bigger than 1000. Call your primes p and q, and write them down.

2) Let n=pq. With a calculator or by hand, calculate n and write it down.

3) With a calculator, a program or by hand, calculate $\phi(n)$.

4) Pick any number **e** such that $\gcd(e, \phi(n))=1$ and $2^e > n$ (you can do this by hand or by writing a program). The number **e** will be used by others to encode secret messages to you. Think "e" for encode.

5) The pair of numbers (n,e) is called your "**public key**". Make your public key 'public': write your group name and public key on the white board. Everybody is allowed to see it. Anybody who wants to write you a secret message can encode it using the public key and 'safely' send it to you.

Part B: Encrypt a secret message to be sent to someone

6) Write down a very brief secret message (maximum 20 characters) that you will send to another student team.

7) Turn your message into numbers by using the number 01 for a, 02 for b and so on up to 26 for z. Use 27 for a space. For example, the message "no problem" becomes

14 15 27 16 18 15 02 12 05 13 or 14152716181502120513

8) Now break the last number into groups of numbers less than n. For this lab break it down to groups of three, our example becomes: 141 527 161 815 021 205 13

9) Select a **public key** (n1, e1) **from another student team** and encrypt the secret message you want to send them: encode each of the number groups in part (8) by finding the remainder when it is raised to the power e1 and divided by n1; i.e. if x is the number, then find x^{e1} mod n1 (you've already written a program that does something like this). In our example, we would begin by finding 141^{e1} mod n1. Write down the resulting blocks of numbers. This gives your encoded message. On the white board, beside the public key, write down your encoded message. Everybody can see it, but, theoretically, only the student team that published its public key can decode it.

Part C: Decode a secret message sent to you

10) Calculate your 'secret decoder': look again at how you constructed *your* public key in Part I, with your choice of **p**, **q** (leading to n), and **e**. Since **e** and $\phi(n)$ are relatively prime, there must be a number **d** so that ed =1 mod $\phi(n)$; i.e., **e** is invertible in $Z\phi(n)$ and **d** is its inverse. Write a computer program to find the smallest d with this property (you may use programs from the first part of the lab). The number **d** is your "**secret decoder**" that will be used for decoding messages and is to be kept **SECRET**. Think "d" for decode.

11) Decode the secret message that has been sent to you by finding the remainder when it is raised to the power **d** (your secret decoder) and divided by **n** (from your public key); i.e. for each block, x, of numbers in the coded message you received, find x^d mod n (you can modify the program from part (9) to do this). Write down the resulting blocks of numbers and see if you can read the message sent to you.

Question 1: How could you send a secret message so that the recipient would know for certain that it came from you?

Question 2: Why is this RSA method for sending encoded secret messages safe (or is it not)?

Lab #7: Exploring Euler's theorem & Introduction to Graphics

- Complete Exercises 1 & 2
- Complete Chapter 15 about graphics

Exercise 1: Introduction to graphics (together)

Create a new vb.net project. Put a picturebox and a button on a form and put the code below under the button to see some of the graphics commands at work. ***Note that the coordinate system on the picture box puts (0,0) at the upper left hand corner***.

```
Dim g As Graphics
Dim sdBrush As SolidBrush = New SolidBrush(Color.Red)
Dim PBlue As New Pen(Color.Blue)
Dim PRed As New Pen(Color.Red)
Dim i, h, w, a, b As Integer
g = PictureBox1.CreateGraphics
```

```
'type the following commands one by one to understand what each of them does
g.DrawLine(PBlue, 23, 56, 200, 300)
g.DrawRectangle(PRed, 23, 56, 20, 30)
g.DrawEllipse(PRed, 150, 150, 250, 50)
g.FillRectangle(sdBrush, 40, 60, 20, 30)
```

```
'what exactly is the following code doing?
h = PictureBox1.Height
w = PictureBox1.Width
For i = 0 To 100
        a = Int(w * Rnd())
        b = Int(h * Rnd())
        g.DrawRectangle(PRed, a, b, 1, 1)
Next i
```

Exercise 2: Exploring Euler's theorem (individually)

Create a new vb.net project to explore Euler's theorem seen in lecture: write a FRIENDLY program that allows a user to enter integers **a** and **n** and then outputs the value of $a^{\phi(n)} \bmod n$, and then use it to explore the theorem. For your code, you may wish to copy some of the vb.net functions you created in Lab #6 and some code from Lab #5.

a) Create the program and test it (ensure that the mathematics is correct using e.g., **n**=5, **a**=2, and **n**=6, **a**=2)
b) Euler's theorem says that if **a** and **n** are relatively prime, then the value of $a^{\phi(n)} \bmod n$ should be 1. Check this theorem using the program you wrote by taking **n** = 7919 (a prime) and different values for **a**.
c) Find values of **a** and **n** so that the value of $a^{\phi(n)} \bmod n$ is not equal to 1

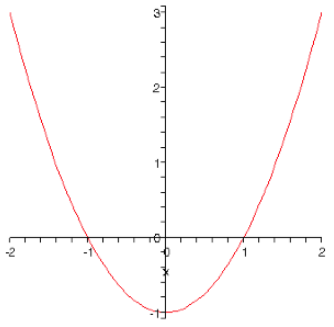
Is the converse of Euler's theorem true? When $a^{\phi(n)} \bmod n$ is 1, does it necessarily imply that **a** and **n** are relatively prime?

Lab #8: Function Graph

This is our first graph in vb.net! Make sure you have completed this exercise before the next lab. Otherwise, you will fall behind as the next lab will build on this exercise.

Graph of a Parabola

Create a new project with two buttons ('Draw the parabola!' and 'End') and one picture box. When clicking on the 'Draw the parabola!' button, the graph of the function $y=x^2-1$ will appear in the picture box. At the end, your graph should look like the following but without the numbers or tick marks on the axes:



a) Create a picture box (600 x 600). Draw the 2 axes in black. Your code will contain something like:

```
Dim g As Graphics
Dim PBlack As New Pen(Color.Black)
g = PictureBox1.CreateGraphics
g.DrawLine(PBlack, ?,?,?,?) 'fill the '?' by the correct vb.net coordinates
g.DrawLine(PBlack, ?,?,?,?) 'fill the '?' by the correct vb.net coordinates
```

b) Draw the origin as a yellow point

c) Draw 5 blue points on the graph of $y= x^2-1$ for x=-2, -1,0,1, and 2.

d) Write a function

```
Function xc(ByVal x As Double) As Integer
```

that changes the x-coordinates for graphing purposes.

e) Write a similar function called yc for the y-coordinates.

f) Draw the graph of the function in red. (***Hint:*** *If (x1,y1) and (x2,y2) are two consecutive points to be graphed, then graph the line joining these two points to get a smoother graph* **or** *draw many points for which the x-values are next to each other.)*

Lab #9: Chaos Experiment Plate-Form Lab

In this lab we create a program that will allow us to systematically explore the dynamical system based on the logistic function next week. The following explains all the steps. It builds on lab 8:

1) Create a new project, called MyChaos, with an interface that contains a button and a picture box. The dimensions of the picture box should be exactly 400 x400.

2) Add a text box called txtInitial, and label it. The value stored in this textbox will be the initial value used in the dynamical system.

3) Add a text box called txtBound, and label it. The value stored in this textbox will be the number of iterations of the dynamical system.

4) Add a text box called txtk. The value stored in this textbox will be the (real) factor k of the logistic function f(x)=kx(1-x). Define k globally, and define a vb.net function:

   Function f(ByVal x As Double) As Double

   which returns the value kx(1-x).

5) Add a text box called txtOutput. The values that are output into this textbox will contain the entire sequence of values generated by the dynamical system. Set the Multiline property to true and then resize the box to make it as large as possible but will still fit on your form. Set the ScrollBars property to Vertical. The steps for creating the output are described next.

6) Write code to graph the parabola f(x)=kx(1-x) when the button is clicked, where k is a real number between 0 and 4 that the user has entered into the textbox. The parabola will always start at the lower left of the picture box and end at the lower right of the picture box.

   - When k=2 the parabola's maximum should be at the middle of the picture box. When k=4 the parabola's maximum should just touch the top of the picture box.
   - Use coordinate change functions for changing the mathematics coordinates to vb.net graphing coordinates (see lab 8)
   - Use a Sub procedure 'Sub DrawFunctionGraph()' to draw the function graph

7) Write a Sub procedure to add the graph of the diagonal y=x

8) Let x_0 be the value the user entered in txtInitial. Write code so that a small red filled circle appears at the point (x_0, x_0) on the screen, when the button is clicked.

9) Define a sequence of numbers by the formula $x_{n+1}=f(x_n)$ for which f(x)=kx(1-x). Let M be the value the user entered in txtBound. Write code that stores the values x_0, x_1,..., x_M in an array. Define the array as global, and have the values computed and saved in the array in a Sub procedure:

   Sub ComputeSequence(ByVal m As Integer)

   where m is the number of terms computed in the sequence, i.e., x0, x1, ..., xm. Also the following M+1 lines of text appear in txtOutput for which the x_i's are the actual numerical values:

```
0          x0
1          x1
```

...	...
M	x_M

10) Add more code to complete the cobweb: when the button is clicked, the following pair of lines is drawn on the screen for every i from i=0 to i=M-1 the line from (x_i, x_i) to (x_i, x_{i+1}) **AND** the line from (x_i, x_{i+1}) to (x_{i+1}, x_{i+1}). This should all be done through a Sub procedure:

```
Sub Cobweb(ByVal M As Integer)
```

And now experiment with different values of k and different values of x_0 . Can you get a general sense of where this dynamical system is well behaved and where it is chaotic? We will explore the amazing behaviour of this system in detail next week.

Additional Programming Exercises

After lab #3

Problem A. The following shows a vb.net code written in a Button1_Click Sub procedure. What will be displayed in Textbox1 after the user clicks on Button1? *(you can easily test your answers by copying and running the code in visual studio)*

a)
```
Dim i,p as Integer
p=2
For i=4 to 11
  p = p+1
Next
Textbox1.text=p
```

b)
```
Dim i,j,p as Integer
p=3
For i=2 to 6
  For j=1 to 3
     p+=i
  Next j
Next i
Textbox1.text=p
```

c)
```
Dim i,sum as Integer
sum=0
For i=1 to 1100
  sum = sum +i
Next
Textbox1.text=sum
```

d)
```
Dim i,sum as Integer
sum=0
For i=1 to 5
  sum = sum +i*i
Next
Textbox1.text=sum
```

e)
```
Dim i,p as Integer
p=1
For i=1 to 9
  p = p*4 mod 15
Next
Textbox1.text=p
```

f)
```
Dim i,j as Integer
Dim p as Long
i=20
p=4
Do while p<5
```

```
            For j=1 to 4
        If i mod 3 =0 Then
           i=i+1
           p=2*p
        Else
           i=i-1
           p=p-1
        End if
    Next j
Loop
Textbox1.text=p
```

g)
```
Dim i,p as Integer
p=7
Do while p > 3
   If p mod 2 = 0 Then
         p+=1
   Else
         p-=3
   End if
Loop
Textbox1.text=p
```

Problem B. write the vb.net code (no need to create a project; write on paper) that computes and displays the result of the following process in a textbox (Textbox1):

a) the sum of the first 100 positive integers, i.e., 1+2+3+.... + 100

b) an integer n is assigned the value 5; if n is even, then add 4 to n and if n is odd, multiply n by 7 and subtract 9; display the result

c) an integer n is assigned the value 8; if n is even, then add 4 to n and if n is odd, multiply n by 7 and subtract 9; display the result

After lab #5

Problem C. The following shows a vb.net code involving a Function procedure. Answer the questions. (you can easily test your answers by copying and running the code in visual studio):

a) A vb.net function is declared in the form:
```
Function f(m as Integer, n as Integer) As Integer
         Dim a,b As  Integer
         a=Math.Max(m,n)
         b=Math.Min(m,n)
         Return a-2*b
End Function
```

In a Button1_Click Sub procedure (the main program), the following statement is written.

```
Textbox1.text= f(6,15)
```

What will be displayed in Textbox1 after the user clicks on Button1?

b) A vb.net function is declared in the form:

```
Function f(m as Integer, n as Integer) As Double
        Dim a,b As  Integer
        a=Math.Max(m,n)
        b=Math.Min(m,n)
        Return Math.Sqrt(3*a-b)
End Function
```

In a Button1_Click Sub procedure (the main program), one wishes to calculate the square root of 3m-n, (m>=n, m and n are integers), assign it to a variable named MyVariable, and display it in a textbox. Write the statement that is missing at line 4:

```
Dim m,n As Integer
m=Textbox1.Text
n=Textbox2.text
????
Textbox3.text = MyVariable
```

<u>Problem D.</u> The following shows a vb.net code written in a Button1_Click Sub procedure. What will be displayed in Textbox1 after the user clicks on Button1? *(you can easily test your answers by copying and running the code in visual studio)*

a)

```
Dim i as Long
Dim F(100) as Long
F(1)=5
For i=2 to 60
  F(i)=F(i-1)+ i
Next
Textbox1.text=F(5)
```

b)

```
Dim i as Long
Dim F(100) as Long
F(1)=5
For i=2 to 60
  F(i)=F(i-1)*2
Next
Textbox1.text=F(6)
```

c)

```
Dim i as Long
Dim F(100) as Long
F(1)=1
F(2)=1
For i=3 to 6
  F(i)=F(i-1)+F(i-2)*2
Next
Textbox1.text=F(5)
```

d)
```
Dim i,j as Integer
Dim p as Long
i=15
p=3
Do while p<5
    For j=1 to 4
        If i mod 3 =0 Then
            i=i+1
            p=2*p
        Else
            i=i-1
            p=p-1
            Exit For
        End if
    Next
Loop
Textbox1.text=p
```

e)
```
Dim i as Long
Dim myArray(100) as Double
myArray(0)=0.5
myArray(1)=0.2
For i=2 to 100
    myArray(i)=(g(myArray(i-1))+g(myArray(i-2))/2
Next
Textbox1.text=myArray(4)
```

And g is defined in the form as

```
Function g(ByVal x As Double) As Double
    Return x*(1-x)
End Function
```

<u>Problem E.</u> Write a vb.Net code that creates each of the following arrays; in each case, call the array myArray:

1. (1,2,3, ..., 100)
2. (1,3,5,7, 9, 11, ...,1001), the array of all positive odd integers < 1002.
3. ($1, a, a^2, a^3, ..., a^{100}$) where a is a positive integer entered by the user in a text box called Textbox1, and a^n is the n*th* power of a in Z_{37}
4. (1,3,6,10, ..., 4005), where the i*th* entry is the sum of the first *i* positive integers.
5. (0, 1, 1, 2, 3, 5, 8, ... , 987), the array of the 17 first terms of the Fibonacci sequence (*by definition, the two first terms are 0 and 1, and all succeeding terms are the sum of the two previous ones*).

Write the vb.Net code that displays a 2-column table in a textbox, called outTextbox, showing, for each array above, the following at each line:

```
i        ith value of myArray
```

After lab #8

Problem F. The following shows a vb.net code involving graphics. Answer the questions.

a) Write an x-coordinate change function for x-values from -5 to 5 and a picture box of size 600 x 600.

b) Write an y-coordinate change function for y-values from -4 to 4 and a picture box of size 600 x 600.

In the following, the coordinate system shows from (0,0) to (10,0) and to (0,10). The picture box is of size 300 x 300.

c) Write a code to draw a rectangle of width 3 and height 4 with bottom-left corner positioned at (5,5).

d) Write a code to draw a line segment from (1,2) to (3,4)

e) Write a code to draw the graph of $y(x) = (1/10)x^2$