Visual Basic

Load the Analysis ToolPak

The Analysis ToolPak is a Microsoft Office Excel add-in program that is available when you install Microsoft Office or Excel. To use it in Excel, however, you need to load it first.

1. Click the **Microsoft Office Button**, and then click **Excel Options**.
2. Click **Add-Ins**, and then in the **Manage** box, select **Excel Add-ins**.
3. Click **Go**.
4. In the **Add-Ins available** box, select the **Analysis ToolPak** check box, and then click **OK**.

   Tip If **Analysis ToolPak** is not listed in the **Add-Ins available** box, click **Browse** to locate it.

   If you get prompted that the Analysis ToolPak is not currently installed on your computer, click **Yes** to install it.
5. After you load the Analysis ToolPak, the **Data Analysis** command is available in the **Analysis** group on the **Data** tab.

NOTE To include Visual Basic for Application (VBA) functions for the Analysis ToolPak, you load the Analysis ToolPak - VBA add-in the same way that you load the Analysis ToolPak. In the **Add-ins available** box, select the **Analysis ToolPak - VBA** check box, and then click **OK**.

Program elements
1) Header line with the name of the function and the parameters
2) Programming lines
3) Closing line (usually inserted by VBA)

START VBA – Developer | Visual Basic

Functions are located – Recently used | Insert function | User defined

To check, you can – Debug| Compile VBA Project

Program 1

```
Function function3(x)
    function3 = x ^ 3 1

End Function
```

Notice, the “1” does nothing

Option Explicit – You have to declare any variables used
Tools | Options | Require Variable Declarations

Declaring variable types – page 439

```
Option Explicit

Function function4(y) As Byte
    'The next line takes a number to the fourth power
    function4 = y ^ 4

End Function
```

NOTE: Anything after an apostrophe is a comment

Variable types

Parameter – External
Temporary – Internal
If it is not a parameter, it is temporary. It is better to do:

```
Dim temporary variable name
```

Multiple parameters

```
Function duoble(X, Y) As Single
    Dim joe
    Dim smo
     joe = X ^ 2
     smo = Y ^ 3
    duoble = joe + smo
End Function
```

Using IF/ESLEIF/ELSE in VBA

Function
    IF statement
        ELSEIF THEN
        ELSEIF THEN
        ELSE
    END IF

An example:

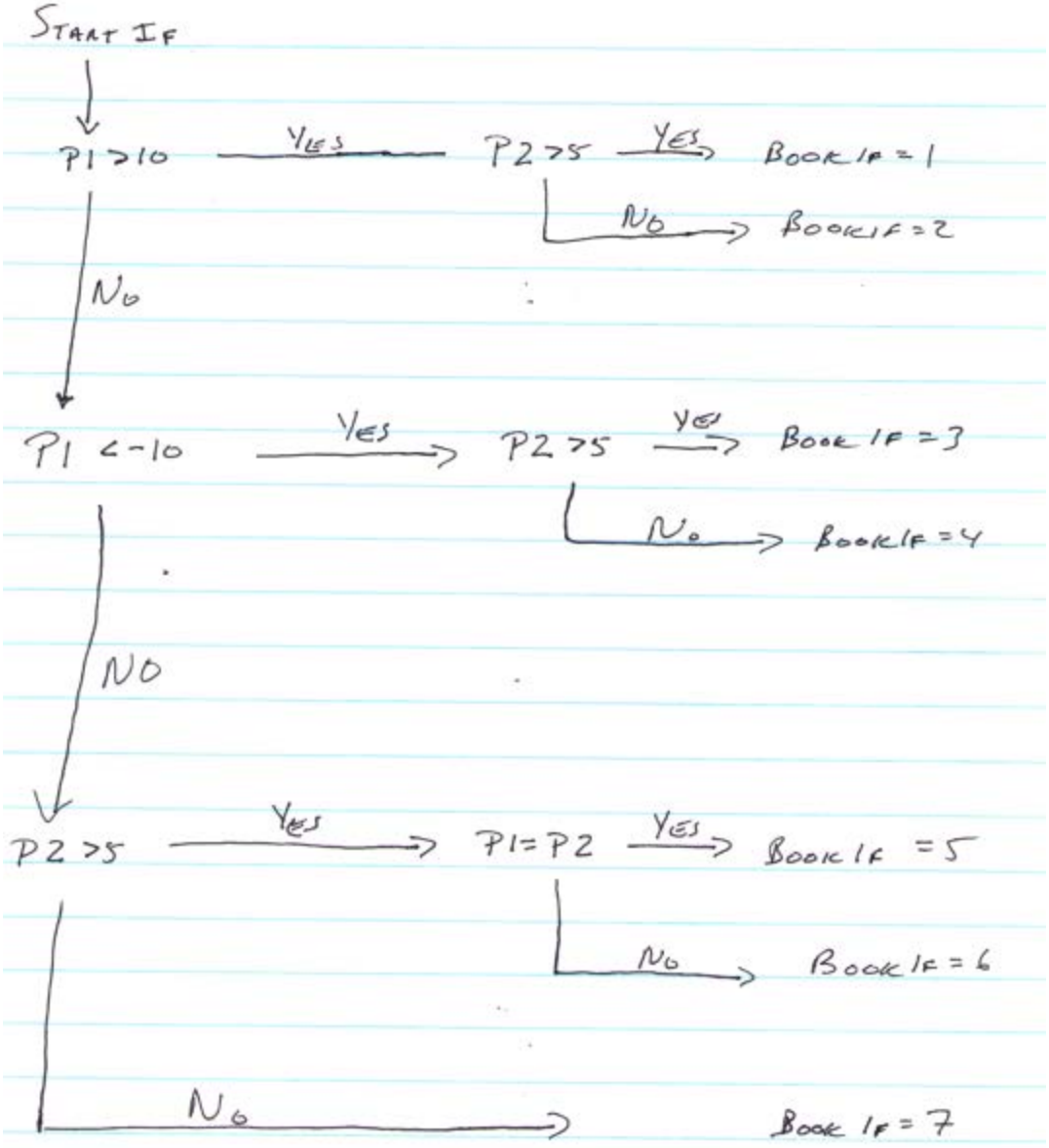
```
Function SimpleIf(parameter)
   SimpleIf = -100
   If parameter <= 25 Then
         SimpleIf = 1
      ElseIf parameter > 25 And parameter <= 50 Then
         SimpleIf = 2
      ElseIf parameter > 50 And parameter <= 75 Then
         SimpleIf = 3
      Else
         SimpleIf = 4
   End If
End Function
```

Suppose we have the following grade system:

Test 1	**Test 2**	**Grade**
>= 85	> = 85	A+
	70-84	A
	< 70	B+
70-85	>=85	A-
	70-85	B
	< 70	B-
<= 70	>=85	C+
	70-85	C
	<= 70	C-

```
Function grade(T1, T2)
   If T1 >= 85 And T2 >= 85 Then
      grade = "A+"
      ElseIf T1 >= 85 And T2 >= 70 Then
         grade = "A"
      ElseIf T1 >= 85 And T2 < 70 Then
         grade = "B+"
      ElseIf T1 >= 70 And T2 >= 85 Then
         grade = "A-"
      ElseIf T1 >= 70 And T2 >= 70 Then
         grade = "B"
      ElseIf T1 >= 70 And T2 < 70 Then
         grade = "B-"
      ElseIf T1 < 70 And T2 >= 85 Then
         grade = "C+"
      ElseIf T1 < 70 And T2 >= 70 Then
         grade = "C"
      Else: grade = "C-"
End If
End Function
```

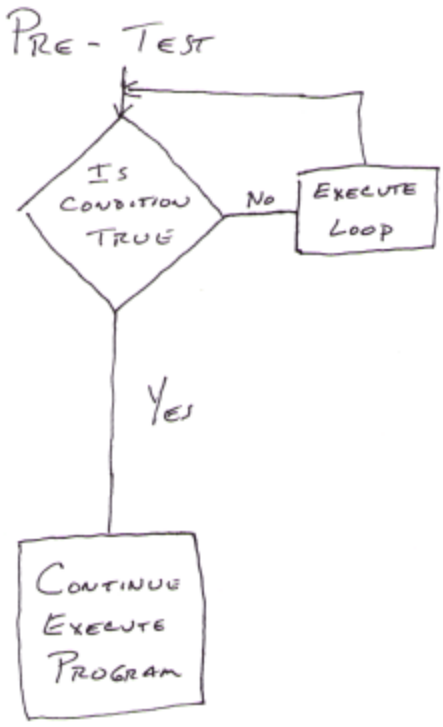
BookIf

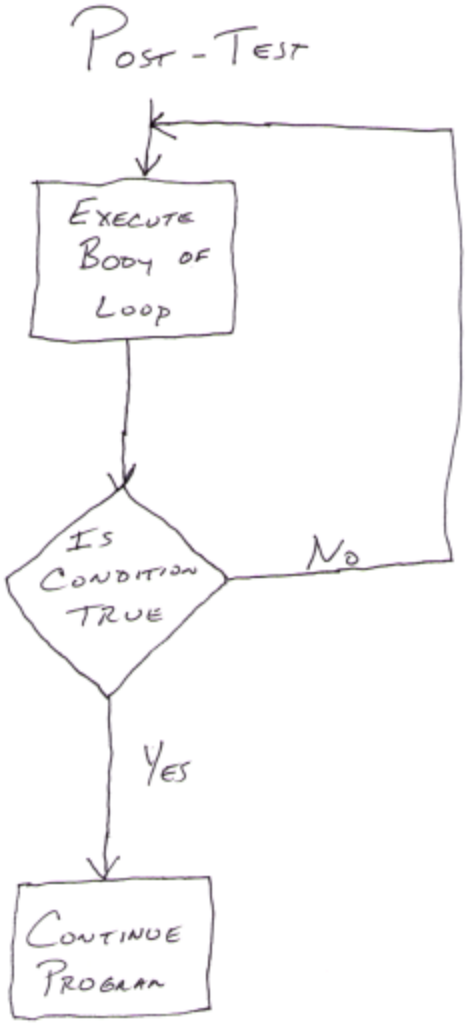

Start If
P1 > 10
Yes
P2 > 5
Yes
BookIf = 1
No
BookIf = 2
No
P1 < -10
Yes
P2 > 5
Yes
BookIf = 3
No
BookIf = 4
NO
P2 > 5
Yes
P1 = P2
Yes
BookIf = 5
No
BookIf = 6
No
BookIf = 7

```
Function BookIf(P1, P2)
   If P1 > 10 Then
      If P2 > 5 Then BookIf = 1 Else BookIf = 2
   ElseIf P1 < -10 Then
      If P2 > 5 Then
         BookIf = 3
      Else
         BookIf = 4
      End If
   Else
      If P2 > 5 Then
         If P1 = P2 Then BookIf = 5 Else BookIf = 6
      Else
         BookIf = 7
      End If
   End If
End Function
```

Loops

Top checking – Pre-test loop – Checks the condition before executing the code
Bottom checking – Post-test – Executes the code one time before testing condition


PRE - TEST
IS CONDITION TRUE
No
EXECUTE LOOP
Yes
CONTINUE EXECUTE PROGRAM


POST - TEST
EXECUTE BODY OF LOOP
IS CONDITION TRUE
No
Yes
CONTINUE PROGRAM

Combination – The number of was R objects can be chosen from N objects when the order does not matter. The lottery is a combination.

$$C = \frac{N!}{(N-R)!\,R!}$$

Permutation – The number of was R objects can be chosen from N objects when the order *does* matter

$$P = \frac{N!}{(N-R)!}$$

Top Down Permutation

```
Function permutation(N, R)
    Dim a
    Dim b
    Dim c
    Dim num
    Dim denom

        a = 1
        num = 1
    'This is a top-down loop that runs N factorial
        Do While a <= N
            num = num * a
            a = a + 1
        Loop

    c = N - R

        b = 1
        denom = 1
    ' This is a top-down loop that runs N - R factorial
        Do While b <= c
            denom = denom * b
            b = b + 1
        Loop

  permutation = num / denom
  If N <= 0 Then
        permutation = -99
  End If
  If N - R > 0 Then
        permutation = -90
  End If
End Function
```

Bottom up Permutation

```
Function permutation1(N, R)
   Dim a
   Dim b
   Dim c
   Dim num
   Dim denom

      a = 1
      num = 1
   'This is a bottom-up loop that runs N factorial
      Do
         num = num * a
         a = a + 1
      Loop While a <= N

   c = N - R

      b = 1
      denom = 1
   ' This is a bottom-up loop that runs N - R factorial
      Do
         denom = denom * b
         b = b + 1
      Loop While b <= c

  permutation1 = num / denom

   If N <= 0 Then
      permutation1 = -99
   End If
   If c < 0 Then
      permutation1 = -90
   End If

End Function
```

Top down Combination

```
Function mycombination(N, R)
   Dim a
   Dim b
   Dim c
   Dim d
   Dim num
   Dim denom1
   Dim denom2
   If N <= 0 Then
      mycombination = -99
   Else
      a = 1
      num = 1
   'This is a top-down loop that runs N factorial
      Do While a <= N
         num = num * a
         a = a + 1
      Loop
   End If
   c = N - R
   If c < 0 Then
      mycombination = -98
   Else
      b = 1
      denom1 = 1
   ' This is a top-down loop that runs N - R factorial
      Do While b <= c
         denom1 = denom1 * b
         b = b + 1
      Loop
   End If
   If R <= 0 Then
      mycombination = -97
   Else
      d = 1
      denom2 = 1
   'This is a top-down loop that runs R factorial
      Do While d <= R
         denom2 = denom2 * d
         d = d + 1
      Loop
   End If
  mycombination = num / (denom1 * denom2)
```

End Function

Bottom up for Combination

```
Function mycombination1(N, R)
    Dim a
    Dim b
    Dim c
    Dim d
    Dim num
    Dim denom1
    Dim denom2
    If N <= 0 Then
       mycombination1 = -99
    Else
       a = 1
       num = 1
    'This is a bottom-up loop that runs N factorial
       Do
          num = num * a
          a = a + 1
       Loop While a <= N
    End If
    c = N - R
    If c < 0 Then
       mycombination1 = -98
    Else
       b = 1
       denom1 = 1
    ' This is a bottom-up loop that runs N - R factorial
       Do
          denom1 = denom1 * b
          b = b + 1
       Loop While b <= c
    End If
    If R <= 0 Then
       mycombination1 = -97
    Else
       d = 1
       denom2 = 1
    'This is a bottom-up loop that runs R factorial
       Do
          denom2 = denom2 * d
          d = d + 1
       Loop While d <= R
    End If
   mycombination1 = num / (denom1 * denom2)
End Function
```

Using Excel's built-in functions

```
Function fvjoe(R, t, ann, pv, typ)

   fvjoe = Application.WorksheetFunction.FV(R, t, ann, pv, typ)

End Function
```

Black-Scholes and the Greeks

```
Option Explicit

Function dOne(S, X, r, sigma, T, q)
   dOne = ((Log(S / X) + (r - q + 0.5 * sigma ^ 2) * T)) / (sigma * Sqr(T))
End Function

Function BSCall(S, X, r, sigma, T, q)
   Dim d1
   d1 = dOne(S, X, r, sigma, T, q)
   BSCall = S * Exp(-q * T) * Application.WorksheetFunction.NormSDist(d1) - X * Exp(-r * T) *
        Application.WorksheetFunction.NormSDist(d1 - sigma * Sqr(T))
End Function

Function BSPut(S, X, r, sigma, T, q)
   Dim d1
   d1 = dOne(S, X, r, sigma, T, q)
   BSPut = -S * Exp(-q * T) * Application.WorksheetFunction.NormSDist(-d1) + X * Exp(-r * T)
        * Application.WorksheetFunction.NormSDist(-d1 + sigma * Sqr(T))
End Function

Function deltaPut(S, X, r, sigma, T, q)
   Dim d1
   d1 = dOne(S, X, r, sigma, T, q)
   deltaPut = Exp(-q * T) * (Application.WorksheetFunction.NormSDist(d1) - 1)
End Function

Function deltaCall(S, X, r, sigma, T, q)
   Dim d1
   d1 = dOne(S, X, r, sigma, T, q)
   deltaCall = Exp(-q * T) * (Application.WorksheetFunction.NormSDist(d1))
End Function
```

```
Function gammaCallPut(S, X, r, sigma, T, q)
   Dim d1, Nprimed1
   d1 = dOne(S, X, r, sigma, T, q)
   Nprimed1 = (1 / ((2 * Application.WorksheetFunction.Pi()) ^ 0.5)) * Exp(-(d1 ^ 2) / 2)
   gammaCallPut = (Nprimed1 * Exp(-q * T)) / (S * sigma * T ^ 0.5)
End Function

Function rhoPut(S, X, r, sigma, T, q)
   Dim d1
   d1 = dOne(S, X, r, sigma, T, q)
   rhoPut = -X * T * Exp(-r * T) * Application.WorksheetFunction.NormSDist(-d1 + sigma *
        Sqr(T))
End Function

Function vegaCallPut(S, X, r, sigma, T, q)
   Dim d1, Nprimed1
   d1 = dOne(S, X, r, sigma, T, q)
   Nprimed1 = (1 / ((2 * Application.WorksheetFunction.Pi()) ^ 0.5)) * Exp(-(d1 ^ 2) / 2)
   vegaCallPut = S * Sqr(T) * Nprimed1 * Exp(-q * T)
End Function

Function thetaPut(S, X, r, sigma, T, q)
   Dim d1, Nprimed1
   d1 = dOne(S, X, r, sigma, T, q)
   Nprimed1 = (1 / ((2 * Application.WorksheetFunction.Pi()) ^ 0.5)) * Exp(-(d1 ^ 2) / 2)
   thetaPut = (-S * Nprimed1 * sigma * Exp(-q * T)) / (2 * Sqr(T)) - q * S *
        Application.WorksheetFunction.NormSDist(-d1) * Exp(-q * T) + r * X * Exp(-r * T) *
        Application.WorksheetFunction.NormSDist(-d1 + sigma * Sqr(T))
End Function
```

```
Function ImpVol(S, X, r, T, q, cm, tol)
   Dim VolEst1, VolEst2, diff, BSC, d1, d2, Nprimed1, Nprimed2, d1prime, d2prime, BSCprime
   VolEst2 = ((2 * Application.WorksheetFunction.Pi() / T) ^ 0.5) * (1 / (S + X)) * (cm - (S - X) /
2 + Sqr((cm - (S - X) / 2) ^ 2 - ((S - X) ^ 2) / Application.WorksheetFunction.Pi()))
   diff = 1
      Do While diff > tol
         VolEst1 = VolEst2
         BSC = BSCall(S, X, r, VolEst1, T, q)
         d1 = dOne(S, X, r, VolEst1, T, q)
         d2 = d1 - VolEst1 * Sqr(T)
         Nprimed1 = (1 / ((2 * Application.WorksheetFunction.Pi()) ^ 0.5)) * Exp(-(d1 ^ 2) / 2)
         Nprimed2 = (1 / ((2 * Application.WorksheetFunction.Pi()) ^ 0.5)) * Exp(-(d2 ^ 2) / 2)
         d1prime = ((Log(S / X) + (r - q) * T) / Sqr(T)) * (-1 / (VolEst1 ^ 2)) + (0.5 * Sqr(T))
         d2prime = d1prime - Sqr(T)
         BSCprime = vegaCallPut(S, X, r, VolEst1, T, q)
         VolEst2 = VolEst1 - (BSC - cm) / BSCprime
         diff = BSC - cm
      Loop
   ImpVol = VolEst2
End Function
```

Assignment #3

Write a VBA program that does the following:

A bank offers different interest rates on deposit accounts based on the size of the periodic deposit (CF). The first deposit is today. Write a future value function **BankFV(CF, r, n)** based on the following interest rates:

Periodic deposit	Interest rate
<= $100.00	r
$100.01 to $499.99	r + 0.5%
$500.00 to $999.99	r + 1.1%
$1,000.00 to $4,999.99	r + 1.7%
>= $5,000	r + 2.1%

Assignment #4

Write a VBA program that calculates a Fibonacci number. Make sure the program uses loops in the calculation.

A Fibonacci number is defined as:
F(N) = F(N – 1) + F(N – 2)

F(0)		= 0
F(1)		= 1
F(2) = F(1) + F(0)	= 1 + 0	= 1
F(3) = F(2) + F(1)	= 1 + 1	= 2
F(4) = F(3) + F(2)	= 2 + 1	= 3
F(5) = F(4) + F(3)	= 3 + 2	= 5
F(6) = F(5) + F(4)	= 5 + 3	= 8
F(7) = F(6) + F(5)	= 8 + 5	= 13