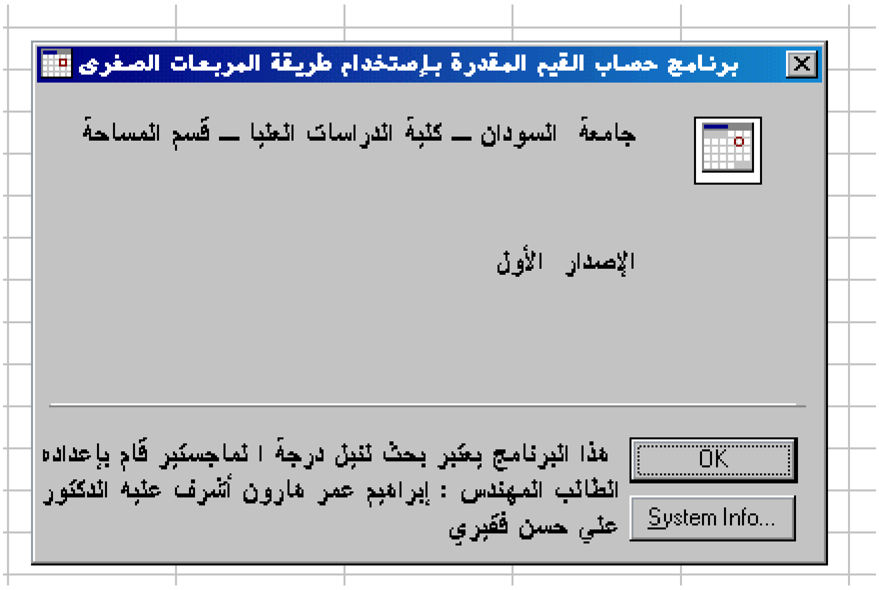
APPENDIX A

THE FRMABOUT FORM :



برنامج حساب القيم المقدرة بإستخدام طريقة المربعات الصغرى

جامعة السودان ــ كلية الدراسات العليا ــ قسم المساحة

الإصدار الأول

هذا البرنامج يعتبر بحث لنيل درجة ا لماجستير قام بإعداده الطالب المهندس : إبراهيم عمر هارون أشرف عليه الدكتور علي حسن فقيري

```
Private Sub cmdOK_Click()
  Unload Me
End Sub
```

```
Dim ignorerow()
Dim ignorediagonal()
Dim ignoredependent()
Dim means()
```

```
Private Sub closewindow_Click()
    Unload frmibrahim
End Sub
```

```
Private Sub Form_DblClick()
    Unload frmibrahim
End Sub
```

```
Private Sub Form_Load()
    flexoutput.ColWidth(0) = 400
    flexoutput.Top = 200
    flexoutput.Left = 200
    flexoutput.Height = Screen.Height - 900
flexoutput.Width = Screen.Width - 400
closewindow.Top = Screen.Height - 600
flexoutput.Rows = FrmInput.labarow * FrmInput.labarow + 7 * FrmInput.labarow
flexoutput.Cols = FrmInput.labacol + 20
Dim temobservation()
Dim temdiagonal()
Dim temdependant()
ReDim temobservation(FrmInput.labarow, FrmInput.labacol)
ReDim temdiagonal(FrmInput.labarow, 0)
ReDim temdependant(FrmInput.labarow, 0)
Dim ibrahim
flexoutput.row = 5
flexoutput.Col = 1
flexoutput.Text = " " & "A"
flexoutput.row = 5
flexoutput.Col = FrmInput.labacol + 1
flexoutput.Text = " " & "B"
flexoutput.row = 5
flexoutput.Col = FrmInput.labacol + 2
flexoutput.Text = " " & "W"
flexoutput.row = 5
flexoutput.Col = FrmInput.labacol + 3
flexoutput.Text = " " & "INV(AẂA)"
flexoutput.row = 5
flexoutput.Col = 2 * FrmInput.labacol + 3
flexoutput.Text = " " & "SIGMA"
For i = 0 To FrmInput.labarow - 1
   For j = 0 To FrmInput.labacol - 1
      flexoutput.row = i + 6
      flexoutput.Col = j + 1
      flexoutput.CellBackColor = &HE0E0E0       ' &HFFFFC0
      flexoutput.Text = " " & Observation(i, j)
   Next j
   flexoutput.row = i + 6
   flexoutput.Col = FrmInput.labacol + 1
   flexoutput.CellBackColor = &HFFFFFF
   flexoutput.Text = " " & dependent(i, 0)
   flexoutput.row = i + 6
   flexoutput.Col = FrmInput.labacol + 2
   flexoutput.CellBackColor = &HE0E0E0
   flexoutput.Text = " " & diagonal(i, 0)
Next i
begin  'Call begin procedure
```

```
  For i = 0 To FrmInput.labarow - 1
      For j = 0 To FrmInput.labacol - 1
          flexoutput.row = i + 6
          flexoutput.Col = FrmInput.labacol + j + 3
          flexoutput.CellBackColor = &HFFFFFF               ' &HC0E0FF
          If i < FrmInput.labacol Then flexoutput.Text = " " & _
          Round(a_w_a_inverse(i, j), 5)
Next j
  Next i
  Dim sigma()
  ReDim sigma(FrmInput.labacol, FrmInput.labacol)
  Dim myvalue
  myvalue = FrmInput.labacol
  Dim count As Integer
  Dim sum
  sum = 0
  For count = 0 To FrmInput.labarow - 1
      If count < FrmInput.labacol Then
          sigma(count, IIf(count = myvalue, 1, count + 1)) = _
          Math.Sqr(a_w_a_inverse(count, count) + a_w_a_inverse _
          (IIf(count = myvalue, 1, count + 1), _
          IIf(count = myvalue, 1, count + 1)) - _
          (2 * a_w_a_inverse(count, IIf(count = myvalue, 1, count + 1))))
          sum = sum + sigma(count, IIf(count = myvalue, 1, count + 1))
      End If
      flexoutput.row = count + 6
      flexoutput.Col = 2 * FrmInput.labacol + 3
      flexoutput.CellForeColor = &HFF0000  ' &HFF&
      flexoutput.CellBackColor = &HFFFFC0
      If count < FrmInput.labacol Then flexoutput.Text = " " & _
      Round(sigma(count, IIf(count = FrmInput.labacol, 1, count + 1)), 7)
      If count = FrmInput.labarow - 1 Then
          flexoutput.Text = " " & Round(sum / FrmInput.labacol, 5)
      End If
  Next
  For i = 0 To FrmInput.labarow
      For j = 0 To FrmInput.labacol
          temobservation(i, j) = Observation(i, j)
      Next j
      temdiagonal(i, 0) = diagonal(i, 0)
      temdependant(i, 0) = dependent(i, 0)
  Next i
  zzz = FrmInput.labarow - 1
```

```
FrmInput.labarow = FrmInput.labarow - 1
FrmInput.labacol = FrmInput.labacol
ReDim means(FrmInput.labarow + 1)
Dim k As Integer
    For k = 1 To FrmInput.labarow + 1
    For i = 0 To FrmInput.labarow
        For j = 0 To FrmInput.labacol - 1
            If i >= zzz Then
                Observation(i, j) = temobservation(i + 1, j)
                diagonal(i, 0) = temdiagonal(i + 1, 0)
                dependent(i, 0) = temdependant(i + 1, 0)
            Else
                Observation(i, j) = temobservation(i, j)
                diagonal(i, 0) = temdiagonal(i, 0)
                dependent(i, 0) = temdependant(i, 0)
            End If
        Next j
    Next i
    ReDim ignorerow(0, FrmInput.labacol - 1)
    ReDim ignorediagonal(0, 0)
    ReDim ignoredependent(0, 0)
    ignorediagonal(0, 0) = temdiagonal(zzz, 0)
    ignoredependent(0, 0) = temdependant(zzz, 0)
    Dim igrow As Integer
    For igrow = 0 To FrmInput.labacol - 1
        ignorerow(0, igrow) = temobservation(zzz, igrow)
    Next igrow
    zzz = zzz - 1
    flexoutput.row = k * (FrmInput.labarow + 1) + 7 + 3 * k
    flexoutput.Col = 0
   '  flexoutput.CellBackColor = &HE0E0E0
     flexoutput.CellFontBold = True
     flexoutput.CellAlignment = 4
    flexoutput.Text = "-" & k & "-"
    flexoutput.Col = 1
    flexoutput.Text = " " & "A"
    flexoutput.Col = FrmInput.labacol + 1
    flexoutput.Text = " " & "B"
    flexoutput.Col = FrmInput.labacol + 2
    flexoutput.Text = " " & "W"
    flexoutput.Col = FrmInput.labacol + 3
    flexoutput.Text = " " & "INV(AẂA)"
    flexoutput.Col = 2 * FrmInput.labacol + 3
flexoutput.Text = " " & "SIGMA"
For i = 0 To FrmInput.labarow - 1
   For j = 0 To FrmInput.labacol - 1
      flexoutput.row = k * (FrmInput.labarow + 1) + i + 8 + 3 * k
      flexoutput.Col = j + 1
      flexoutput.CellBackColor = &HE0E0E0       ' &HFFFFC0
      flexoutput.Text = " " & Observation(i, j)
   Next j
   flexoutput.row = k * (FrmInput.labarow + 1) + i + 8 + 3 * k
   flexoutput.Col = FrmInput.labacol + 1
   flexoutput.CellBackColor = &HFFFFFF
   flexoutput.Text = " " & dependent(i, 0)
   flexoutput.row = k * (FrmInput.labarow + 1) + i + 8 + 3 * k
   flexoutput.Col = FrmInput.labacol + 2
   flexoutput.CellBackColor = &HE0E0E0
   flexoutput.Text = " " & diagonal(i, 0)
Next i
Dim mmm As Integer
flexoutput.row = k * (FrmInput.labarow + 1) + i + 8 + 3 * k
flexoutput.Col = 0
flexoutput.CellFontName = "Arabic Transparent"
flexoutput.CellFontSize = 15
flexoutput.CellForeColor = vbRed
flexoutput.CellAlignment = 7
flexoutput.Text = "*"
For mmm = 0 To FrmInput.labacol - 1
   flexoutput.Col = mmm + 1
   flexoutput.Text = " " & ignorerow(0, mmm)
   flexoutput.CellBackColor = &HE0E0E0
Next mmm
flexoutput.Col = mmm + 1
flexoutput.Text = " " & ignoredependent(0, 0)
flexoutput.CellBackColor = &HE0E0E0
flexoutput.Col = mmm + 2
flexoutput.Text = " " & ignorediagonal(0, 0)
flexoutput.CellBackColor = &HE0E0E0
Dim backignore
For backignore = 1 To FrmInput.labacol + 1
   flexoutput.Col = mmm + 2 + backignore
   flexoutput.CellBackColor = &HE0E0E0
Next
begin  'Call begin procedure
For i = 0 To FrmInput.labarow - 1
```

```
    For j = 0 To FrmInput.labacol - 1
        flexoutput.row = k * (FrmInput.labarow + 1) + i + 8 + 3 * k
        flexoutput.Col = FrmInput.labacol + j + 3
        flexoutput.CellBackColor = &HFFFFFF
        If i < FrmInput.labacol Then flexoutput.Text = " " & _
        Round(a_w_a_inverse(i, j), 5)
        '''flexoutput.Col = 2 * FrmInput.labacol + j + 6
        '''If i < FrmInput.labacol Then
          '''flexoutput.CellBackColor = &HFFFFC0
          ''' flexoutput.Text = " " & a_w_a(i, j)
        '''End If
    Next j
Next i
On Error Resume Next
sum = 0
For count = 0 To FrmInput.labarow - 1
    If count < FrmInput.labacol Then
        sigma(count, IIf(count = myvalue, 1, count + 1)) = _
        Math.Sqr(a_w_a_inverse(count, count) + a_w_a_inverse _
        (IIf(count = myvalue, 1, count + 1), _
        IIf(count = myvalue, 1, count + 1)) - _
        (2 * a_w_a_inverse(count, IIf(count = myvalue, 1, count + 1))))
        sum = sum + sigma(count, IIf(count = myvalue, 1, count + 1))
    End If
    flexoutput.row = k * (FrmInput.labarow + 1) + count + 8 + 3 * k
    flexoutput.Col = 2 * FrmInput.labacol + 3
    flexoutput.CellForeColor = &HFF0000  ' &HFF&
    flexoutput.CellBackColor = &HFFFFC0
    If count < FrmInput.labacol Then flexoutput.Text = " " & _
    Round(sigma(count, IIf(count = FrmInput.labacol, 1, count + 1)), 5)
    If count = FrmInput.labarow - 1 Then
        flexoutput.row = k * (FrmInput.labarow + 1) + count + 9 + 3 * k
        flexoutput.CellForeColor = &HFF0000  ' &HFF&
        flexoutput.CellBackColor = &HFFFFC0
        flexoutput.CellFontUnderline = True
        flexoutput.CellAlignment = 5
        means(k) = Round(sum / FrmInput.labacol, 5)
        flexoutput.Text = means(k)
    End If
Next
Erase Observation
Erase dependent
Erase diagonal
```

```
Next k
For i = 1 To FrmInput.labacol + 2
    flexoutput.ColWidth(i) = 750
Next
'Dim xxx As Integer
summeans = 0
For i = 1 To FrmInput.labarow + 1
    summeans = summeans + means(i)
Next
avg_summeans = summeans / (FrmInput.labarow + 1)
 flexoutput.Col = 7
 lastrow = (k - 1) * (FrmInput.labarow + 1) + count + 12 + 3 * (k - 1)

flexoutput.row = lastrow
For i = 1 To 11
    flexoutput.Col = i
    flexoutput.CellBackColor = vbYellow
Next
 flexoutput.Col = 6
 flexoutput.Text = "Since :"
' flexoutput.Col = 7
' flexoutput.Text = "All :"
 flexoutput.row = lastrow + 1

 Dim STR
 Dim mycounter
 mycounter = 0
 For i = 1 To FrmInput.labarow + 1
    If means(i) > avg_summeans Then
        mycounter = mycounter + 1
        flexoutput.row = lastrow + mycounter
        flexoutput.Col = 7
        flexoutput.Text = " (" & i & ")"
        flexoutput.Col = 8
        flexoutput.Text = " " & means(i)
    End If
 Next
For i = 1 To 11
    flexoutput.row = lastrow + mycounter + 1
    flexoutput.Col = i
    flexoutput.CellBackColor = vbYellow
Next
    flexoutput.row = lastrow + mycounter + 1
```

```
        flexoutput.Col = 8
        flexoutput.Text = " > " & Round(avg_summeans, 5)
        flexoutput.Col = 9
        flexoutput.Text = "we dismiss"
        flexoutput.Col = 10
        flexoutput.Text = "it"
    End Sub
```

General module :

```
'Option Explicit Statement Used at module level to force explicit
'declaration of all variables in that module
'If used, the Option Explicit statement must appear in a module before
'anyprocedures.
'When Option Explicit appears in a module, you must explicitly declare
'all variables using the Dim, Private, Public, ReDim, or Static
'statements. If you attempt to use an undeclared variable name, an error
'occurs atcompile time.
'If you don't use the Option Explicit statement, all undeclared variables
'are of Variant type unless the default type is otherwise specified with
'a Deftype statement.

Option Explicit
   Public frmreadmatrix_visible As Boolean 'By default the value of this variable =
false ,
                                   'it becometrue when we load frmreadmatrix form to
                                   'read the value of matrixes from frminput form
   Public txtreadidxgotfocus ' The variable that we save the index value
                        ' of the txtinput control (text class) in it
   Public sr1, sc1, er1, ec1 'the variable that i save dimentions of matrix so when one
                        'of the text in the frmreadmatrix set focus we see some cell area in
the grid
                        'of the frminput form with another back color
   Public row As Integer, Col As Integer
   Public f_cal '(f statistics) the variable that we save the calculation value to compare
              'with tabulation value of F table
   Public beta(0 To 20, 0) 'the least square stimater (of Beta for example)
   Public F_TAB, df_e, df_r, df_t, sign, decision, level ' the values of :
   'F_TAB      F from table
   'df_e       degree of freedom (error)
   'df_t       degtee of freedom (total)
   Public dirty, rowcount, colcount As Integer
   'dirty      to test if the file where we save value has change
   'rowcount   to read the rows count of matrix in the file we save data
   'colcount   to save the cols count of matrix in the file we save data
   Public i, j, sr, er, sc, ec, ff, co, ro, r, z As Integer
   Public dependent(0 To 30, 0), ba(0 To 30, 0), wa(0 To 30, 0) 'the dependent
variable
   Public Observation(0 To 20, 0 To 20), diagonal(0 To 30, 0), weight(0 To 20, 0 To
20)
   'the observations or the independent vareables
```

```
    Public observation_2(0, 20), dependent_2(0, 0), weight_2(0, 0)
    'the variables when we read another values
    '-------------------
    Public A2_X1_B2
    Public input_matrex(), b()
    Public n, M, k, Substitute_by
    Public w_a(), a_w_a(), w_d(), a_w_y()
    Public a_w_a_inverse(), A_N_A2()
    Public A_N2(), N2_A()
    Public OBSERVATION_2_BETA()
    Public N1_INVERSE_OBSERVATION_2()
    Public NAWANAAXB()
    Public beta_2()
    Public A_B()
    Public w_b()
    Public c()
    Public cx()
    Public cxhat()
    Public SST, SSR, SSE, MSR, MSE, f 'the variable of the ANOVA TABLE
    'VARIABLE       STAND FOR
    'SST            sum square total
    'SSR            sum square regression
    'SSE            sum squre error
    'MSR             mean squre regrssion
    'MSE             mean squre error
    'f          f-statistics = MSR/MSE
    Public coefficient_of_determination   'The coefficient of determination
```

```
Sub main()
    'We want this application to start without any form initially loaded , so we .
    'create this Sub procedure called Main in this a standard module.


                         '**************************
                         '** program begin here**
                         '**************************

10: frmreadmatrix_visible = False
    ' I set the working directory to the directory containing this application.
20: ChDir App.Path
    ' Application starts here (Load event of Startup form).
30: FrmInput.Show 'the frminput is the startup form
End Sub
```

```
Sub resize_form() 'Occurs when an object is first displayed or when the window state
of
                'an object changes. '(For example, a form is maximized, minimized, or
                'restored.)
On Error Resume Next
   If FrmInput.WindowState = 1 Then Exit Sub
   If FrmInput.tbToolBar.Visible Then
      FrmInput.piccalculate.Top = FrmInput.tbToolBar.Height
      FrmInput.piccalculate.Left = 0
      FrmInput.piccalculate.Width = FrmInput.Width
      FrmInput.FlexInput.Height = _
                           FrmInput.ScaleHeight - _
                           FrmInput.tbToolBar.Height - _
                           FrmInput.StatusBar.Height - _
                           FrmInput.piccalculate.Height
      FrmInput.FlexInput.Width = FrmInput.ScaleWidth
      FrmInput.FlexInput.Top = _
                        FrmInput.tbToolBar.Height + _
                        FrmInput.piccalculate.Height
   Else
      FrmInput.piccalculate.Top = 0
      FrmInput.piccalculate.Left = 0
      FrmInput.FlexInput.Height = _
                           FrmInput.ScaleHeight - _
                           FrmInput.StatusBar.Height - _
                           FrmInput.piccalculate.Height
      FrmInput.FlexInput.Width = FrmInput.ScaleWidth
      FrmInput.FlexInput.Top = FrmInput.piccalculate.Height
      FrmInput.piccalculate.Width = FrmInput.Width
   End If
End Sub
```

```
Sub calculate_beta(co, ro)
'to calculate the estimated value of first reading values
   Dim w
   w = 0
   For i = 0 To ro - 1
      For r = 0 To ro - 1
         w = w + a_w_a_inverse(r, i) * a_w_y(r, 0)
      Next r
      beta(i, 0) = w
      w = 0
```

```
    Next i
End Sub
```

```
Sub EXTENTION(co, ro)
    Dim z, r, zzz
    ReDim A_N2(0, co)
    ReDim N2_A(0, co)
    ReDim OBSERVATION_2_BETA(0, 0)
    ReDim N1_INVERSE_OBSERVATION_2(co, 0)
    ReDim NAWANAAXB(co, 0)
    ReDim beta_2(co, 0)
    ReDim c(0, co)
    ReDim cx(co, co)
    ReDim cxhat(co, co)
    For i = 0 To co - 1
        z = 0
        zzz = 0
        r = 0
        For j = 0 To co - 1
            z = z + observation_2(0, r) * a_w_a_inverse(r, i)
            zzz = zzz + a_w_a_inverse(r, i) * observation_2(0, r)
            r = r + 1
        Next j
        A_N2(0, i) = z
        N2_A(0, i) = zzz
    Next i

    z = 0
    r = 0
    For i = 0 To co - 1
        z = z + A_N2(0, r) * observation_2(0, r)
        r = r + 1
    Next i

    Dim a
    a = z + 1 / weight_2(0, 0)
    Dim M
    M = 1 / a

    z = 0
    r = 0
    For i = 0 To co - 1
        z = z + observation_2(0, r) * beta(r, 0)  'calculate A2X1^
        r = r + 1
    Next i
    OBSERVATION_2_BETA(0, 0) = z

    Dim II
    II = dependent_2(0, 0)
    A2_X1_B2 = OBSERVATION_2_BETA(0, 0) - dependent_2(0, 0) '0.029 '  -
    'Print A2_X1_B2
    Dim W_ANA_AX_B
    W_ANA_AX_B = A2_X1_B2 * 1 / a

    ' CALCULATE N1' * A2
    For i = 0 To co - 1
        z = 0
        r = 0
        For j = 0 To co - 1
            z = z + a_w_a_inverse(i, r) * observation_2(0, r)
            r = r + 1
        Next j
        N1_INVERSE_OBSERVATION_2(i, 0) = z   'N1A2
    Next i

    For i = 0 To co - 1
        NAWANAAXB(i, 0) = N1_INVERSE_OBSERVATION_2(i, 0) *
W_ANA_AX_B
    Next i
    For i = 0 To co - 1
        beta_2(i, 0) = beta(i, 0) - NAWANAAXB(i, 0)
    Next i

    For i = 0 To co
        c(0, i) = N2_A(0, i) * M
    Next i

    'to obtain the new covarianc matrix :
    For i = 0 To co - 1
        z = 0
        r = 0
        For j = 0 To co - 1
            z = c(0, i) * A_N2(0, r)
            cx(i, j) = z   'N1A2
            r = r + 1
        Next j
    Next i

    For i = 0 To co - 1
        For j = 0 To co - 1
            'the product of subtract
            cxhat(i, j) = Round(a_w_a_inverse(i, j), 7) - Round(cx(i, j), 7)
        Next j
    Next i
    End Sub
```

```
Sub begin()
    row = FrmInput.labarow
    Col = FrmInput.labacol
    For i = 0 To row - 1
        For j = 0 To row - 1
            weight(i, j) = 0
        Next j
    Next i

    If frmreadmatrix.Check2.Value = 1 Then
        For i = 0 To row - 1
            For j = 0 To row - 1
                weight(i, i) = diagonal(i, 0)
            Next j
        Next i
    Else
        For i = 0 To row - 1
            For j = 0 To row - 1
                weight(i, j) = 0
                weight(i, i) = 1
            Next j
        Next i
    End If
    ReDim a_w_a_inverse(row, Col)
    Call weight_observation(row, Col)
    For i = 0 To Col - 1
        For j = 0 To Col - 1
            a_w_a_inverse(i, j) = b(i, j)
        Next j
    Next i

    Call weight_dependent(row, Col)
    Call calculate_beta(row, Col)
    '-------------------------------------------------------
    If frmreadmatrix.Check4.Value = 0 Then
        anova  ' the output
        Exit Sub
    End If
    If frmreadmatrix.Check3.Value = 0 Then
        ' If weight_2(0, 0) = "" Then   ' InputBox("ENTER WEIGHT_2(1,1)" &
weight_2(0, 0))
        weight_2(0, 0) = 1
    End If
    Call EXTENTION(Col, row)
    anova
End Sub
```

```
Sub weight_observation(ro, co)
    ReDim w_a(ro, co)
    ReDim a_w_a(co, co)
    ReDim input_matrex(0 To co, 0 To co)
    Dim X
    For i = 0 To ro - 1
        For j = 0 To co - 1
            For r = 0 To ro - 1
             '  If Observation(r, j) = "" Then MsgBox "قم بإدخال  قيمة في الخانة " & "(" & r
& "," & j & ")"
                If Observation(r, j) = "" Then MsgBox "البرنامج لا ي قبل  قيم مجهولة": Exit
Sub
                z = z + weight(i, r) * Observation(r, j)
            Next r
            w_a(i, j) = z
            z = 0
        Next j
    Next i

    ' A*W*A *************************
    For i = 0 To co - 1
        For j = 0 To co - 1
            For r = 0 To ro - 1
                z = z + Observation(r, i) * w_a(r, j)
            Next r
            a_w_a(i, j) = z
            z = 0
        Next j
    Next i

    For i = 0 To co - 1
        For j = 0 To co - 1
            input_matrex(i, j) = a_w_a(i, j)
        Next j
    Next i
    calculate_inverse (co)
End Sub
```

```
Sub weight_dependent(ro, co)
    Dim z
    ReDim a_w_y(co, 0)
    ReDim w_d(ro, 0)
    For i = 0 To ro - 1
        For r = 0 To ro - 1
            z = z + weight(i, r) * dependent(r, 0)
        Next r
        w_d(i, 0) = z
        z = 0
    Next i

    z = 0
    For i = 0 To co - 1
        For r = 0 To ro - 1
            z = z + Observation(r, i) * w_d(r, 0)
        Next r
        a_w_y(i, 0) = z
        z = 0
    Next i
End Sub
```

```
Sub anova()
On Error GoTo nizoerror:
    ReDim A_B(FrmInput.labacol, 0)
    ReDim w_b(FrmInput.labarow, 0)
    Dim z
    z = 0
    For i = 0 To FrmInput.labarow - 1
        For j = 0 To FrmInput.labarow - 1
            'If frmreadmatrix.Check2.Value = 1 Then we take weight
            ' as we enter it else we take weight as identity matrix
            z = z + weight(i, j) * dependent(j, 0)
        Next j
        w_b(i, 0) = z
        z = 0
    Next i
    z = 0
    'to calculate SST

    For i = 0 To FrmInput.labarow - 1
    z = z + dependent(i, 0) * w_b(i, 0)
    Next i
    SST = z

    z = 0
    For i = 0 To FrmInput.labacol - 1
        For j = 0 To FrmInput.labarow - 1
            z = z + Observation(j, i) * w_b(j, 0)
        Next j
        A_B(i, 0) = z
        z = 0
    Next i

    'TO CALCULATE SSR
    z = 0
    For i = 0 To FrmInput.labacol - 1
        z = z + beta(i, 0) * A_B(i, 0)
    Next i

    'The following values summarizes the sums of squares , degrees of freedom of
    'associated chi distributions . In the mean squares , which are sums
    'of squares divided by degrees of freedom , it also shows calculation
    'of the numerator and denominator of F . And then the calculation of
    'F itself is shown .
    'Thus the analysis of variance table is simply a convenient summary of the
    'steps involved in calculation an F-statistics .

    SSR = Round(z, 3) 'To calculate the sum of square of regression
    SSE = SST - SSR  'To calculate the sum of square of residual
    MSR = SSR / FrmInput.labacol 'To calculate the mean square of regression
    MSE = SSE / (FrmInput.labarow - FrmInput.labacol) 'To calculate the mean square
      of residual

    ' TO CALCULATE THE Coefficient Of Determination
```

```
    coefficient_of_determination = SSR / SST ' to calculate the coefficient of
determination

    frmOutput.flexoutput.TextMatrix(0, 0) = "S.O.V" 'source of variance
    frmOutput.flexoutput.TextMatrix(1, 0) = "REGRESSION"
    frmOutput.flexoutput.TextMatrix(2, 0) = "RESIDUAL"
    frmOutput.flexoutput.TextMatrix(3, 0) = "TOTAL"

    frmOutput.flexoutput.TextMatrix(0, 1) = "D.F" ' degree of freedom
    frmOutput.flexoutput.TextMatrix(1, 1) = STR(FrmInput.labacol)
    frmOutput.flexoutput.TextMatrix(2, 1) = STR(FrmInput.labarow -
FrmInput.labacol)
    frmOutput.flexoutput.TextMatrix(3, 1) = STR(FrmInput.labarow)

    frmOutput.flexoutput.TextMatrix(0, 2) = "S.S" 'sum of square
    frmOutput.flexoutput.TextMatrix(1, 2) = STR(SSR)
    frmOutput.flexoutput.TextMatrix(2, 2) = STR(SSE)
    frmOutput.flexoutput.TextMatrix(3, 2) = STR(SST)

    frmOutput.flexoutput.TextMatrix(0, 3) = "M.S" ' mean square
    frmOutput.flexoutput.TextMatrix(1, 3) = STR(MSR)
    frmOutput.flexoutput.TextMatrix(2, 3) = STR(MSE)

    frmOutput.flexoutput.TextMatrix(0, 4) = "F" 'The (F) calclated value
    f_cal = MSR / MSE ' f_cal is the (F) calculated value
    frmOutput.flexoutput.TextMatrix(1, 4) = STR(Round(f_cal, 3))

    frmOutput.flexoutput.TextMatrix(0, 5) = "F-TABULATED"

    df_e = FrmInput.labarow - FrmInput.labacol ' df_e is the degree of freedom of
error
    df_r = FrmInput.labacol ' df_r is the degree of freedom of regression
    F_TAB = f_tabulated(df_e, df_r) ' F_TAB is the F value that given from table
    frmOutput.flexoutput.TextMatrix(1, 5) = STR(F_TAB)

    If F_TAB > f_cal Then
        sign = "<"
        decision = "accept"
    Else
        sign = ">"
        decision = "reject"
    End If
    If frmreadmatrix.Combo1.Text = "0.05" Then
```

```
        level = "5%"
    End If
    If frmreadmatrix.Combo1.Text = "0.01" Then
        level = "1%"
    End If
    If frmreadmatrix.Combo1.Text = "0.025" Then
        level = "2.5%"
    End If
    Exit Sub
nizoerror:
    MsgBox "Please if you received this message call : " & vbNewLine & _
    " Ibrahim Omar Haroon (mobile: 012134708) ", vbCritical, "Problem"
End Sub
```

```
Sub calculate_inverse(n As Integer)
'this procedure calculate the inverse matrix
    ReDim b(0 To n, 0 To 2 * n)
    M = 2 * n
    For i = 0 To n - 1
        For j = 0 To n - 1
            b(i, j + n) = 0
            b(i, j) = input_matrex(i, j)
        Next j
        b(i, i + n) = 1
    Next i

    For k = 0 To n - 1
        If k = n - 1 Then GoTo Jump1
        M = k
        For i = k + 1 To n - 1
            If Abs(b(i, k)) > Abs(b(M, k)) Then M = i
        Next i
        If M = k Then GoTo Jump1
        For j = k To 2 * n - 1
            Substitute_by = b(k, j)
            b(k, j) = b(M, j)
            b(M, j) = Substitute_by
        Next j

Jump1:
        For j = k + 1 To 2 * n - 1
            If b(k, k) = 0 Then GoTo Jump2
            b(k, j) = b(k, j) / b(k, k): GoTo Jump3
```

```
Jump2:
            b(k, j) = 0
Jump3:
        Next j
        If k = 0 Then GoTo Jump4
        For i = 0 To k - 1
            For j = k + 1 To 2 * n - 1
                b(i, j) = b(i, j) - b(i, k) * b(k, j)
            Next j
        Next i
        If k = n - 1 Then GoTo Jump5
Jump4:
        For i = k + 1 To n - 1
            For j = k + 1 To 2 * n - 1
                b(i, j) = b(i, j) - b(i, k) * b(k, j)
            Next j
        Next i
    Next k

Jump5:
    For i = 0 To n - 1
        For j = 0 To n - 1
            b(i, j) = b(i, j + n)
        Next j
    Next i

End Sub
```

```
Sub erase_values()
    'The erase statement Reinitializes the elements of fixed-sizearrays and
    'releases dynamic-array storage space.
    Erase input_matrex(), b(), w_a(), a_w_a(), w_d(), a_w_y()
    Erase a_w_a_inverse(), A_N_A2(), A_N2(), N2_A()
    Erase OBSERVATION_2_BETA(), N1_INVERSE_OBSERVATION_2(),
NAWANAAXB()
    Erase beta_2(), A_B(), w_b(), c(), cx(), cxhat()
    Set SST = Nothing 'The Nothing keyword is used to disassociate an
    'object variable from an actual object. Use the Set statement to
    'assign Nothing to an object variable.
    ' dependent(0 To 30, 0), ba(0 To 30, 0), wa(0 To 30, 0)
    ' Observation(0 To 20, 0 To 20), diagonal(0 To 30, 0), weight(0 To 20, 0 To 20)
    ' observation_2(0, 20), dependent_2(0, 0), weight_2(0, 0)
    If sr1 <> "" Then
```

```
        For i = sr1 To er1
            For j = sc1 To ec1
                FrmInput.FlexInput.row = i
                FrmInput.FlexInput.Col = j
                FrmInput.FlexInput.CellBackColor = FrmInput.FlexInput.BackColor
            Next j
        Next i
    End If
End Sub
```