LISTING PROGRAM

1. Menu Utama

Private Sub Abt_Click()
    frmAbout.Show
End Sub
Private Sub exi_Click()
    End
End Sub
Private Sub Hlp_Click()
    frmHelp.Show
End Sub
Private Sub MDIForm_Activate()
    Timer1.Enabled = True
End Sub
Private Sub MDIForm_Load()
    Timer1.Enabled = True
End Sub
Private Sub Timer1_Timer()
    If Timer1.Enabled = True Then
        Line1.BorderColor = &H0&
        Shape1.FillColor = vbRed
        Line10.BorderColor = &H0&
        Shape2.FillColor = vbBlue
        Line9.BorderColor = &H0&
        Shape3.FillColor = vbYellow
        Shape4.FillColor = vbGreen
        Shape5.FillColor = vbCyan
        Shape6.FillColor = vbBlack
        Shape7.FillColor = vbRed
        Shape8.FillColor = vbGreen
        Timer1.Enabled = False
        Timer2.Enabled = True
    Else
        Shape1.FillColor = &HFFFFFF
        Shape2.FillColor = &HFFFFFF
        Shape3.FillColor = &HFFFFFF
        Shape4.FillColor = &HFFFFFF
        Shape5.FillColor = &HFFFFFF
        Shape6.FillColor = &HFFFFFF
        Shape7.FillColor = &HFFFFFF
        Shape8.FillColor = &HFFFFFF
        Timer1.Enabled = True
    End If
End Sub

Private Sub Timer2_Timer()
If Timer2.Enabled = True Then
    Line1.BorderColor = &HFF&
    Shape1.FillColor = &HFFFFFF
    Line10.BorderColor = &HFF&
    Shape2.FillColor = &HFFFFFF
    Line9.BorderColor = &HFF&
    Shape3.FillColor = &HFFFFFF
    Shape4.FillColor = &HFFFFFF
    Shape5.FillColor = &HFFFFFF
    Shape6.FillColor = &HFFFFFF
    Shape7.FillColor = &HFFFFFF
    Shape8.FillColor = &HFFFFFF
    Timer2.Enabled = False
    Timer1.Enabled = True
Else
    Shape1.FillColor = vbRed
    Shape2.FillColor = vbBlue
    Shape3.FillColor = vbYellow
    Shape4.FillColor = vbGreen
    Shape5.FillColor = vbCyan
    Shape6.FillColor = vbBlack
    Shape7.FillColor = vbRed
    Shape8.FillColor = vbGreen
    Timer2.Enabled = True
End If
End Sub

Private Sub TSP_Click()
    frmTSP.Show
End Sub

2. TSP

Dim OLDbestI As Long
Dim GA As New SimplyGA
Dim NumberOfGenes As Long
Dim CICLES As Long
Dim P() As tPoint
Dim I As Long
Dim G As Long
Private Sub cmdCancel_Click()
    txtNP = ""
    txtNP.SetFocus
End Sub
Private Sub Command1_Click()
    Dim GAgeneration As Long
    Dim ChartX
    Dim Cities As String
Dim QueryAddress As String
Dim S As String
Dim S2() As String
Dim KM As Single
Dim Tempo As String
Dim TS() As String
Dim ValTempo As Single
Init_GA
Dim NOI
Dim NOG As Long
Dim BesT As Long
OLDbestI = 1
ChartX = 0
' Stop
NOI = GA.NumberOfIndivid
NOG = GA.NumberOfGenes(1)
For CICLES = 1 To Val(txtCICL)
    For I = 1 To NOI
        If GA.IsIndiChanged(I) Then
            D = 0
            For G = 1 To NOG - 1
                D = D + Distance(P(GA.getGENE(I, G)), P(GA.getGENE(I, G + 1)))
            Next G
            D = D + Distance(P(GA.getGENE(I, 1)), P(GA.getGENE(I, NOG)))
            GA.IndiFitness(I) = D
        End If
    Next I
    GA.COMPUTEGENES
    GEN.Text = GA.StatGeneration
    ACC.Text = GA.StatAccop
    MUT.Text = GA.StatMutations
    NEWr.Text = GA.StatNEWs
    BFIT.Text = GA.GeneratBestFit
    gAVG.Text = GA.GenerationAvgFit
    BesT = GA.GeneratINDEXBestFit
    If OLDbestI <> BesT Then
        Me.Cls
        For G = 1 To NOG - 1
            P1 = GA.getGENE(BesT, G)
            P2 = GA.getGENE(BesT, G + 1)
            Me.Line (P(P1).x, P(P1).Y)-(P(P2).x, P(P2).Y), vbBlack
            Me.Circle (P(P1).x, P(P1).Y), 4, vbBlue
        ' Me.Print P1
        Next G
        P1 = GA.getGENE(BesT, 1)
        P2 = GA.getGENE(BesT, NOG)
        Me.Line (P(P1).x, P(P1).Y)-(P(P2).x, P(P2).Y), vbBlack
        Me.Circle (P(P2).x, P(P2).Y), 4, vbBlue
OLDbestI = Best

End If

GAgeneration = GA.StatGeneration

If (GAgeneration Mod 50) = 0 Then
    ChartX = ChartX + 1
    MSChart.RowCount = ChartX 'GaGeneration
    MSChart.DataGrid.SetData ChartX, 1, GA.GeneratBestFit, Flag
    MSChart.DataGrid.SetData ChartX, 2, GA.GenerationAvgFit, Flag
End If

If (GAgeneration Mod 100) = 0 Then DoEvents

Next CICLES

End Sub

Private Sub Command2_Click()

CICLES = Val(txtCICL)

End Sub

Private Sub Command3_Click()

InitLocations

End Sub

Private Sub Form_Load()

Randomize Timer

MSChart.ColumnCount = 2
MSChart.RowCount = 0

InitLocations

End Sub

Public Sub InitLocations()

Me.Cls

ReDim P(Val(txtNP))

For ip = 1 To UBound(P)
    P(ip).x = 20 + Rnd * 320
    P(ip).Y = 160 + Rnd * 350
    Me.Circle (P(ip).x, P(ip).Y), 4, vbBlue

Next

End Sub

Sub Init_GA()

NumberOfGenes = UBound(P)

txtNI = NumberOfGenes * 5

txtNI = "30"

NumberOfGenes * NumberOfGenes / 2

GA.INIT Val(txtNI), NumberOfGenes, 1, NumberOfGenes, _
    0.01, 0.1, enRank, 0.01, False, SonToWorst, TSProblem, INFO, 1000 ' 250

End Sub
3. **Modul-1**

Public Type tPoint
   x As Single
   Y As Single
End Type

Public Function Distance(P1 As tPoint, P2 As tPoint) As Single
   Dim dX As Single
   Dim dY As Single
   dX = P1.x - P2.x
   dY = P1.Y - P2.Y
   Distance = Sqr(dX * dX + dY * dY)
End Function

4. **Class-1**

Option Explicit
Dim enRandO
Dim enWheel
Dim enRank
Dim SonToWorst
Dim SonToNewINDI
Dim SonToRndINDI
Dim SonToParent
Dim SwapG
Dim CrossG

#End If

Public Enum Enum_SelMode
   enRandO = 0
   enWheel = 1
   enRank = 2
End Enum

Public Enum Enum_ReproductMode
   SonToRndINDI = 0
   SonToWorst = 1
   SonToParent = 2
   SonToNewINDI = 3
End Enum

Public Enum Enum_CrossMode
   SwapG = 0
   CrossG = 1
   TSPProblem = 2
End Enum
Private Type tStat
    NofACC As Long
    NofMUT As Long
    NofNEW As Long
    NofGEN As Long
End Type

Private Type tInd
    NofG As Long
    Gene() As Long
    Fitness As Double
    Wheel As Double
    NetOutPut As Double
    FreeForSon As Boolean
    IsChanged As Boolean
End Type

Event TESTINDIevent(Individ As Long)

Private SonMode As Enum_ReproductMode
Private CrossMode As Enum_CrossMode
Private NofI As Integer
Private INDI() As tInd
Private gValueMax As Long
Private gValueMin As Long
Private MutProbProp As Boolean
Private MutProb As Single
Private MutRate As Single
Private BestFit As Double
Private GenerationBestFit As Double
Private GenerationINDEXBestFit As Long

Private Statistic As tStat
Private pSelectionMode As Enum_SelMode
Private pMutateBestFit As Boolean
Private ReprodXGeneration As Long
Private BestGENE() As Long
Private LookForDisaster As Long
Private AVGfit As Double
Private INFO As TextBox
Public Property Let INDINetOutPut(indiv, Val As Double)
    INDI(indiv).NetOutPut = Val
End Property
Public Property Get pLookForDisaster() As Long
    pLookForDisaster = LookForDisaster
End Property
Public Property Get IsIndiChanged(ind As Long) As Boolean 'since perv generation
    IsIndiChanged = INDI(ind).IsChanged
End Property
Public Property Get Son_Mode() As Enum_ReproductMode
Son_Mode = SonMode
End Property
Public Property Get Cross_Mode() As Enum_CrossMode
Cross_Mode = CrossMode
End Property
Public Property Let Son_Mode(sMode As Enum_ReproductMode)
SonMode = sMode
End Property
Public Property Let Cross_Mode(cMode As Enum_CrossMode)
CrossMode = cMode
End Property
Public Property Get pSelection_Mode() As Enum_SelMode
pSelection_Mode = pSelectionMode
End Property
Public Property Let pSelection_Mode(sMode As Enum_SelMode)
pSelectionMode = sMode
End Property
Public Property Get pMutateBestToo() As Single
pMutateBestToo = pMutateBestFit
End Property
Public Property Get pMutationProb() As Single
pMutationProb = MutProb
End Property
Public Property Get pMutationRate() As Single
pMutationRate = MutRate
End Property
Public Property Get StatAccop() As Long
StatAccop = Statistic.NofACC
End Property
Public Property Get StatGeneration() As Long
StatGeneration = Statistic.NofGEN
End Property
Public Property Get StatMutations() As Long
StatMutations = Statistic.NofMUT
End Property
Public Property Get StatNEWs() As Long
StatNEWs = Statistic.NofNEW
End Property
Public Property Get Get_gValueMin() As Double
Get_gValueMin = gValueMin
End Property
Public Property Get Get_gValueMax() As Double
Get_gValueMax = gValueMax
End Property
Public Property Get GeneratBestFit() As Double
GeneratBestFit = GenerationBestFit
End Property
Public Property Get GeneratINDEXBestFit() As Double
GeneratINDEXBestFit = GenerationINDEXBestFit
End Property
Public Property Get GenerationAvgFit() As Double
GenerationAvgFit = AVGfit
End Property
Public Property Let IndiFitness(Individ, FitnessValue As Double)
INDI(Individ).Fitness = FitnessValue
End Property
Public Property Get IndiFitness(Individ) As Double
IndiFitness = INDI(Individ).Fitness
End Property
Public Property Get NumberOfIndivid() As Long
NumberOfIndivid = NofI
End Property
Public Property Get NumberOfGenes(indiv) As Long
NumberOfGenes = INDI(indiv).NofG
End Property
Private Function GfnCreateRandomIndi(NofGene, gvMIN, gvMAX) As tInd
Dim G As Long
Dim Gpos As Long
GfnCreateRandomIndi.NofG = NofGene
GfnCreateRandomIndi.Fitness = 1E+99
ReDim GfnCreateRandomIndi.Gene(NofGene)
If CrossMode <> TSProblem Then
    For G = 1 To NofGene
        GfnCreateRandomIndi.Gene(G) = fnRND(gvMIN, gvMAX, True)
    Next
Else 'TSP
    For G = 1 To NofGene
        GfnCreateRandomIndi.Gene(G) = 0
    Next
    For G = 1 To NofGene
        Do
            Gpos = fnRND(1, NofGene, True)
        Loop While GfnCreateRandomIndi.Gene(Gpos) <> 0
        GfnCreateRandomIndi.Gene(Gpos) = G
    Next
End If
GfnCreateRandomIndi.IsChanged = True
End Function
Private Function fnRND(Min, Max, DoRound As Boolean)
fnRND = Rnd * (Max - Min) + Min
If DoRound Then fnRND = Round(fnRND)
'Debug.Print "fnRND___ ", min, max, fnRND
End Function
Public Sub INIT(NofIndi, NofGene, gvMIN, gvMAX, _
    Mutate_Prob, Mutate_Rate, _
    SelMode As Enum_SelMode, PercNewSonXGeneration, _
    MutBest As Boolean, _
    ReproductMode As Enum_ReproductMode, _
    aCrossMode As Enum_CrossMode, TextINFO As TextBox, _
    Optional LookForDisasterEvery As Long)
Dim I As Long
Dim S As String
Dim NOG As Long
Dim G As Long
ReprodXGeneration = Round(PercNewSonXGeneration * NofIndi)
If ReprodXGeneration < 1 Then ReprodXGeneration = 1
'Stop
'Mutation Prob=0 means proportional
MutProbProp = IIf(Mutate_Prob = 0, True, False)
Set INFO = TextINFO
NofI = NofIndi
MutProb = Mutate_Prob
MutRate = Mutate_Rate
gValueMax = gvMAX
gValueMin = gvMIN
pSelectionMode = SelMode
pMutateBestFit = MutBest
SonMode = ReproductMode
CrossMode = aCrossMode
BestFit = 1E+99
ReDim BestGENE(NofGene)
ReDim INDI(NofIndi)
For I = 1 To NofIndi
    INDI(I) = GfnCreateRandomIndi(NofGene, gvMIN, gvMAX)
    MutateIndi I, True
    MutateIndi I, True
Next
If LookForDisasterEvery = 0 Then LookForDisasterEvery = 2 ^ 30 '1000000000
LookForDisaster = LookForDisasterEvery
Statistic.NofACC = 0
Statistic.NofMUT = 0
Statistic.NofNEW = 0
Statistic.NofGEN = 0
S = "Genetic Algorithm Initialized! " & vbCrLf & vbCrLf
S = S + "Current Parameters:
S = S + "Number of Individuals  " & vbTab & NofI & vbCrLf
S = S + "Number of Gene x Indiv" & vbTab & NofGene & vbCrLf
S = S + "Mutation Probability " & vbTab & Mutate_Prob * 100 & "%" & vbCrLf
S = S + "Mutation Rate (gene)  " & vbTab & Mutate_Rate * 100 & "%" & vbCrLf
S = S + "Minim Gene Value      " & vbTab & gValueMin & vbCrLf
Universitas Sumatera Utara
S = S + "Max Gene Value     " & vbTab & gValueMax & vbCrLf
S = S + "Parent Selection Mode " & vbTab & pSelectionMode & vbCrLf
S = S + "Can even Mutate Best I" & vbTab & MutBest & vbCrLf
S = S + "Placement of Son     " & vbTab & SonMode & vbCrLf
S = S + "Cross Mode          " & vbTab & CrossMode & vbCrLf
S = S + "Test Disaster Every " & vbTab & LookForDisaster & vbCrLf

'MsgBox S, vbInformation, "GA created!"
End Sub

Public Sub ReplaceIdenticalINDI(INFOtext As TextBox)
Dim I As Long
Dim j As Long
Dim NOG As Long
Dim G As Long
Dim T As Integer
Dim iden() As Long
Dim S As String
Dim PervNumOfNew
'DebugPrintPop PP
PervNumOfNew = Statistic.NofNEW
NOG = INDI(1).NofG
For I = 1 To NofI - 1
    For j = I + 1 To NofI
        G = 1
        Q1:
        If G <= NOG Then If INDI(I).Gene(G) = INDI(j).Gene(G) Then G = G + 1:
        GoTo Q1
        If G > NOG Then
            Statistic.NofNEW = Statistic.NofNEW + 1
            INDI(I) = GfnCreateRandomIndi(NOG, gValueMin, gValueMax)
        End If
    Next j
Next I
If Len(INFOtext) > 10000 Then INFOtext.Text = ""
S = S & " Look for Disaster. New Random Individ Generated = " & _
    Statistic.NofNEW - PervNumOfNew & " of " & NofI & vbCrLf
    Statistic.NofNEW & vbCrLf & vbCrLf
INFOtext.Text = INFOtext.Text & S
INFOtext.SelStart = Len(INFOtext.Text)
'DebugPrintPop pp
End Sub

Private Sub MutateIndi(Individ, MutateBest As Boolean)
Dim OLD
Dim MR
Dim G As Long
Dim G2 As Long

Universitas Sumatera Utara
Dim T As Integer
Dim Times As Integer
Dim NOG As Long
Dim POS As Long
INDI(Individ).IsChanged = True
If CrossMode <> TSPProblem Then
    If Not (MutateBest) Then If GenerationINDEXBestFit = Individ Then Exit Sub
    Statistic.NofMUT = Statistic.NofMUT + 1
    MR = MutRate
    With INDI(Individ)
        """cambia ""
        G = fnRND(1, INDI(Individ).NofG, True)
        OLD = .Gene(G)
        MR = fnRND(-MutRate / 2, MutRate / 2, False)
        OLD = OLD + MR * (gValueMax - gValueMin)
        If OLD < gValueMin Then OLD = gValueMin
        If OLD > gValueMax Then OLD = gValueMax
        Gene(G) = OLD
        .Fitness = 1E+99
        .Wheel = 0
    End With
End If 'TSP
Else 'TSP
    If Not (MutateBest) Then If GenerationINDEXBestFit = Individ Then Exit Sub
    Statistic.NofMUT = Statistic.NofMUT + 1
    NOG = INDI(Individ).NofG
    With INDI(Individ)
        POS = fnRND(1, NOG, True)
        G2 = .Gene(POS)
        For T = POS To NOG - 1
            .Gene(T) = .Gene(T + 1)
        Next
        POS = fnRND(1, NOG, True)
        For T = NOG To POS + 1 Step -1
            .Gene(T) = .Gene(T - 1)
        Next
        .Gene(POS) = G2
    End With
End If
End Sub
Private Sub MutateAll()
Dim I As Long
Dim Mp As Single
'Mp = MutProb * (100 / NoFl)
Mp = MutProb
For I = 1 To NoFl

'MUTATION
    '  Stop
    If Rnd < Mp Then
    '    Stop
    '    Stop
        MutateIndi I, pMutateBestFit
    End If
Next I

End Sub
Private Sub FITNESSTest()
Dim I As Long
Dim IndexOverALLBest As Long
Dim IndexGenerationBest As Long
Dim OverALLBest As Boolean
Dim GenerationBest As Boolean
OverALLBest = False
GenerationBest = False
GenerationBestFit = 1E+99
AVGfit = 0
For I = 1 To NofI
    INDI(I).FreeForSon = True
    If INDI(I).Fitness < BestFit Then
        BestFit = INDI(I).Fitness
        OverALLBest = True
        IndexOverALLBest = I
    End If
    If INDI(I).Fitness < GenerationBestFit Then
        GenerationBestFit = INDI(I).Fitness
        GenerationBest = True
        IndexGenerationBest = I
    End If
    AVGfit = AVGfit + INDI(I).Fitness
Next I
AVGfit = AVGfit / NofI
EvalFitness IndexOverALLBest, IndexGenerationBest, _
            OverALLBest, GenerationBest, INFO
End Sub

Private Sub TEST_INDI(Individ As Long)
With INDI(Individ)
    .Fitness = .NetOutPut
End With
Stop
End Sub
Private Sub EvalFitness(IndexOverALLBest As Long, IndexGenerationBest As Long, OverALLBest As Boolean, GenerationBest As Boolean, INFOtext As TextBox)

Dim S As String
Dim GG As Long
Dim Gc As Long
If OverALLBest Then
    With INDI(IndexOverALLBest)
        S = "G " & Statistic.NofGEN & " Indi " & IndexOverALLBest & " f(" & .Fitness & ") " & vbTab & vbTab
        For GG = 1 To .NofG
            S = S & .Gene(GG) & vbTab
            BestGENE(GG) = .Gene(GG)
        Next
    End With
    INFOtext.Text = INFOtext.Text & S & vbCrLf
    INFOtext.SelLength = 1
    INFOtext.SelStart = Len(INFOtext.Text)
    DebugPrintPop
    If BestFit = 0 Then DebugPrintPop: MsgBox "Solution FOUND!!! " & vbCrLf & _
        "Generation: " & Statistic.NofGEN & " Individ: " & IndexOverALLBest
    Stop
End If
End Sub

Public Sub DebugPrintPop()

Dim S As String
Dim I As Long
Dim G As Long
Debug.Print
Debug.Print "POPULATION"
For I = 1 To NofI
    S = "I" & I & " FIT:" & INDI(I).Fitness & vbTab & vbTab
    For G = 1 To INDI(I).NofG
        S = S & vbTab & INDI(I).Gene(G)
    Next
    Debug.Print S
Next
End Sub

Private Sub CreaWheel()

Dim Sum As Double
Dim Sum2 As Double
Dim Sum3 As Double
Dim MinF As Double
Dim MaxF As Double
Dim I As Long
MinF = 1E+99
MaxF = -1E+99
Sum = 0
For I = 1 To NofI
    With INDI(I)
        Sum = Sum + .Fitness
        If .Fitness < MinF Then MinF = .Fitness
        If .Fitness > MaxF Then MaxF = .Fitness
    End With
Next
MaxF = MaxF + 1
Sum2 = 0
For I = 1 To NofI
    Sum2 = Sum2 + (MaxF - INDI(I).Fitness)
Next
Sum3 = 0
'Debug.Print "WHEEL______________"
For I = 1 To NofI
    With INDI(I)
        Sum3 = Sum3 + 100 * (MaxF - .Fitness) / Sum2
        Debug.Print "fitness " & I, .Fitness, MaxF - .Fitness, 100 * (MaxF - .Fitness) / Sum2, Sum3
        .Wheel = Sum3
    End With
Next
'Stop
End Sub

Private Sub CreaRankWheel()
Dim Sum As Double
Dim Sum2 As Double
Dim Sum3 As Double
Dim MinF As Double
Dim MaxF As Double
Dim I As Long
MinF = 1E+99
MaxF = -1E+99
Sum = 0
For I = 1 To NofI
    With Indi(I)
        Sum = Sum + I '.Fitness
        If .Fitness < MinF Then MinF = .Fitness
        If .Fitness > MaxF Then MaxF = .Fitness
    End With
Next
MinF = 1
MaxF = NofI
MaxF = MaxF + 1
Sum2 = 0
For I = 1 To Nofi
    Sum2 = Sum2 + (MaxF - Indi(I).Fitness)
    Sum2 = Sum2 + MaxF - I
Next
Stop

Sum3 = 0
For I = 1 To Nofi
    With INDI(I)
        ' Sum3 = Sum3 + 100 * (MaxF - .Fitness) / Sum2
        Sum3 = Sum3 + 100 * (MaxF - I) / Sum2
        .Wheel = Sum3
    End With
Next
'Stop
End Sub

Private Function GfnFindParent() As Long
    Dim R As Double
    Dim Wm As Double
    Dim I As Long
    R = fnRND(0, 100, False)
    For I = 1 To Nofi
        Wm = IIf(I = 1, 0, INDI(I - 1).Wheel)
        If R >= Wm And R < INDI(I).Wheel Then GfnFindParent = I: Exit For
    Next
    Debug.Print "findp " & r, "Parent " & GfnFindParent
    Stop
    Stop
End Function

Private Sub SelectParents(ByRef P1 As Long, ByRef P2 As Long)
    Select Case pSelectionMode
        Case enWheel
            WHEEL
            CreaWheel
            Do
                P1 = GfnFindParent
                P2 = GfnFindParent
                Loop While P1 = P2
        Case enRank
            QuickSortFitness 1, Nofi
            RankWheel
            CreaRankWheel
            Do
                P1 = GfnFindParent
                P2 = GfnFindParent
                If P1 = 0 Or P2 = 0 Then Stop
    End Case
End Sub
Loop While P1 = P2
Case enRandO
    Do
        P1 = fnRND(1, NofI, True)
        P2 = fnRND(1, NofI, True)
    Loop While P1 = P2
End Select
End Sub

Public Sub Accoppia(Par1 As Long, Par2 As Long, pSonMode As Enum_ReproductMode, pCrossMode As Enum_CrossMode)
    'MATE
    Dim NewIndi As tInd
    Dim NewIndi2 As tInd
    Dim G As Long
    Dim G2 As Long
    Dim G3 As Long
    Dim WORST As Long
    Dim Par As Long
    Dim I2 As Long
    Dim tmpNofG As Long
    Dim VV2 As Integer
    Dim V1 As Integer
    Dim V2 As Integer
    Dim oo As Integer
    Dim NOG As Long
    Dim SON As Long
    Statistic.NofACC = Statistic.NofACC + 1
    NewIndi.Fitness = 1E+99
    NewIndi.NofG = INDI(Par1).NofG
    NewIndi.Wheel = 0 'da calcolare
    NewIndi.IsChanged = True
    Debug.Print
    Debug.Print "ACCOPIA"
    With INDI(Par1)
    End With
    'With INDI(Par2)
    'End With
    'crossover
    Select Case pCrossMode
        Case SwapG
            For G = 1 To INDI(Par1).NofG
                If Rnd < 0.5 Then
NewIndi.Gene(G) = INDI(Par1).Gene(G)
Else
   NewIndi.Gene(G) = INDI(Par2).Gene(G)
End If
Next
Case CrossG
CROSS
  tmpNofG = INDI(Par1).NofG
  For I2 = 1 To tmpNofG
     NewIndi.Gene(I2) = INDI(Par1).Gene(I2)
  Next
  For VV2 = 1 To 2
     'Stop
     G = fnRND(1, tmpNofG - 1, True)
     G2 = fnRND(G, G + tmpNofG * 0.2, True) '%% of genes max cross
     If G2 > tmpNofG Then G2 = tmpNofG
     'Debug.Print "               G replaced from " & G & " to " & G2 & " (of " &
     tmpNofG & ")"
     "'For I2 = 1 To G - 1
     "   NewIndi.Gene(I2) = Indi(Par1).Gene(I2)
     "'Next
     For I2 = G To G2 - 1
       NewIndi.Gene(I2) = INDI(Par2).Gene(I2)
     Next
     'For I2 = G2 To tmpNofG
     '   NewIndi.Gene(I2) = Indi(Par1).Gene(I2)
     'Next
  Next
Next VV2
Case TSProblem
NewIndi = EdgeRecombinationCrossover(Par1, Par2)
GoTo skip
'The parents:
'31|1111 (CABDEF)
'11|1211 (ABCEFD)
'The children:
'11|1111 (CABDEF)
'31|1211 (ABEDFC)
Dim O1() As Integer
Dim O2() As Integer
ReDim O1(INDI(Par1).NofG)
ReDim O2(INDI(Par1).NofG)
TSPBuildO O1, Par1
TSPBuildO O2, Par2
NOG = INDI(Par1).NofG
Do
   V1 = fnRND(1, NOG, True) '2
   V2 = fnRND(1, NOG, True)
Loop While V2 < V1
V2 = NOG
  If Rnd < 0.5 Then
    For oo = V1 To V2
      O1(oo) = O2(oo)
    Next oo
  Else
    For oo = V1 To V2
      O2(oo) = O1(oo)
    Next oo
    O1 = O2 ""
  End If

NewIndi = TSPBuildNewIndiFromO(O1)
NewIndi2 = NewIndi
GoTo skip
For G = 1 To NOG
  If NewIndi.Gene(G) = 1 Then Exit For
Next
G3 = G - 1
For G = 1 To NOG
  G2 = G3 + G: If G2 > NOG Then G2 = G2 - NOG
  NewIndi.Gene(G) = NewIndi2.Gene(G2)
Next G
skip:
End Select

'Stop
Select Case pSonMode
  Case SonToWorst
    " replace Worst
    WORST = GfnGetWORSTindi
    INDI(WORST) = NewIndi
  Case SonToNewINDI
    " ADDtoPop
    ADDIndi NewIndi
  Case SonToRndINDI
    " Replace rnd
    INDI(fnRND(1, NofI, True)) = NewIndi
  Case SonToParent
    " Replace one of 2 parents
    Par = IIf(Rnd < 0.5, Par1, Par2)
    INDI(Par) = NewIndi
End Select
End Sub
Private Sub ADDIndi(ind As tInd)
NofI = NofI + 1
ReDim Preserve INDI(NofI)
INDI(NofI) = ind
End Sub
Public Function GfnGetWORSTindi() As Long
'private
Dim MaxF As Double
Dim I As Long
MaxF = -1E+99
For I = 1 To NofI
    With INDI(I)
        If I <> GenerationINDEXBestFit Then"useless
            Stop
        If .FreeForSon Then
            If .Fitness > MaxF Then MaxF = .Fitness: GfnGetWORSTindi = I
        End If
    End With
    End If
Next
INDI(GfnGetWORSTindi).FreeForSon = False
End Function
Public Sub COMPUTEGENES()
Dim II As Long
Dim BF
Dim Parent1 As Long
Dim Parent2 As Long
Dim S As String
Dim Bool As Boolean
'Stop
Dim V As Long
Statistic.NofGEN = Statistic.NofGEN + 1
'Stop
For II = 1 To NofI
    INDI(II).IsChanged = False
Next
'MUTATE
MutateAll
'Stop
'FITNESS TEST
FITNESStest
Stop
For V = 1 To ReprodXGeneration '* NofI
    'GETPARENTS
    SelectParents Parent1, Parent2
    'REPRODUCTION
    Accoppia Parent1, Parent2, SonMode, CrossMode
Next V
'MutateAll
Stop
If Statistic.NofGEN / LookForDisaster = Statistic.NofGEN \ LookForDisaster Then
    Stop
End If

If GeneratBestFit / GenerationAvgFit > 0.9995 Then ReplaceIdenticalINDI INFO > 0.997
If GenerationBestFit = GenerationAvgFit Then ReplaceIdenticalINDI INFO
Debug.Print GeneratBestFit / GenerationAvgFit
If MutProbProp Then
    MutProb = (GenerationBestFit / GenerationAvgFit) / 10 ^ 5
End If
Stop
End Sub

Public Function getGENE(Individ As Long, Gene As Long) As Long
    getGENE = INDI(Individ).Gene(Gene)
End Function

Public Sub SetGENE(Individ As Long, Gene, Valu As Long)
    INDI(Individ).Gene(Gene) = Valu
End Sub

Public Sub KillIndi(Individ As Long)
    Dim II As Long
    For II = Individ To NofI - 1
        INDI(II) = INDI(II + 1)
    Next II
    NofI = NofI - 1
    ReDim Preserve INDI(NofI)
End Sub

public Sub Save_POP(Optional Filename = "POP.txt")
    Dim II  As Long
    Dim GG As Long
    Open App.Path & ":" & Filename For Output As 1
    Print #1, "Num of INDIs"
    Print #1, NofI
    Print #1, "Num of GENEs (x indi)"
    Print #1, INDI(1).NofG
    Print #1, "Min Max Gene Value"
    Print #1, gValueMin
    Print #1, gValueMax
    Print #1, "Mutation Prob"
    Print #1, Replace(MutProb, ",", ".")
    Print #1, "Mutation Rate"
    Print #1, Replace(MutRate, ",", ".")
    Print #1, "Selection Mode"
    Print #1, pSelectionMode
Print #1, "ReprodXGeneration"
Print #1, ReprodXGeneration
Print #1, "Son       Mode"
Print #1, SonMode
Print #1, "Cross      Mode"
Print #1, CrossMode
Print #1, "Mutate Best"
Print #1, CInt(pMutateBestFit)
Print #1, "INDEX BEST INDI"
Print #1, GenerationINDEXBestFit
Print #1, "LookForDisasterEvery"
Print #1, LookForDisaster
For II = 1 To NofI
    Print #1, "---------------INDI " & II
    For GG = 1 To INDI(II).NofG
        Print #1, INDI(II).Gene(GG)
    Next GG
Next II
Close 1
End Sub
Public Sub Load_POP(Optional Filename = "POP.txt")
Dim II  As Long
Dim GG As Long
Dim S As String
Open App.Path & ":" & Filename For Input As 1
Input #1, S
Input #1, NofI
ReDim INDI(0 To NofI)
Input #1, S
Input #1, INDI(1).NofG
ReDim BestGENE(1 To INDI(1).NofG)
"Stop
Input #1, S
Input #1, gValueMin
Input #1, gValueMax
Input #1, S
Input #1, MutProb
Input #1, S
Input #1, MutRate
Input #1, S
Input #1, pSelectionMode
Input #1, S
Input #1, ReprodXGeneration
Input #1, S
Input #1, SonMode
Input #1, S
Input #1, CrossMode
Input #1, S
Input #1, S
If S = "-1" Then pMutateBestFit = True Else: pMutateBestFit = False
Input #1, S
Input #1, GenerationINDEXBestFit
Input #1, S
Input #1, LookForDisaster
For II = 1 To NofI
   Input #1, S
   ReDim INDI(II).Gene(1 To INDI(1).NofG)
   INDI(II).NofG = INDI(1).NofG
   For GG = 1 To INDI(II).NofG
      Input #1, INDI(II).Gene(GG)
      INDI(II).Fitness = 1E+16
   Next GG
Next II
Close 1
INFO = ":" End Sub
Public Sub RandomALLIndi()
Dim I As Long
For I = 1 To NofI
   INDI(I) = GfnCreateRandomIndi(INDI(I).NofG, gValueMin, gValueMax)
   MutateIndi I, True
Next I
BestFit = 1E+99
GenerationBestFit = 1E+99
GenerationINDEXBestFit = 1
End Sub
Private Sub TSPBuildO(ByRef O, ind As Long) 'USELESS
Dim busy() As Boolean
Dim G As Long
Dim NOG As Long
Dim NB As Integer
Dim POS As Integer
NOG = INDI(ind).NofG
ReDim busy(NOG)

With INDI(ind)
   O(1) = .Gene(1)
   busy(.Gene(1)) = True
   'Debug.Print O(1) & " Gval=" & .Gene(1)
   For G = 2 To NOG
End Sub
NB = 0
For POS = 1 To .Gene(G)
    If busy(POS) = False Then NB = NB + 1
Next
O(G) = NB
    Debug.Print NB & "   Gval=" & .Gene(G)
baby(.Gene(G)) = True
If NB = 0 Or .Gene(G) = 0 Then Stop
Next
End With
End Sub

Private Function TSPBuildNewIndiFromO(ByRef O) As tInd 'USELESS
Dim tINDI As tInd
Dim busy() As Boolean
Dim G As Long
Dim NOG As Long
Dim NB As Integer
Dim POS As Integer
NOG = INDI(1).NofG
ReDim busy(NOG)
With tINDI
    ReDim .Gene(NOG)
    .NofG = NOG
    .Fitness = 1E+99
    .Wheel = 0 'da calcolare
    .Gene(1) = O(1)
    'Debug.Print O(1) & "   g" & .Gene(1)
baby(.Gene(1)) = True

    For G = 2 To NOG
        NB = 0
        POS = 0
        Do
            POS = POS + 1
            If POS > NOG Then POS = 1
            If busy(POS) = False Then NB = NB + 1
        Loop While NB <> O(G)
        .Gene(G) = POS
        '   Debug.Print O(G) & "   g" & POS
        busy(.Gene(G)) = True
        If O(G) = 0 Or POS = 0 Then Stop
    Next
'Debug.Print "---"
End With
TSPBuildNewIndiFromO = tINDI
End Function
Sub SortByFitness()
Dim I1 As Long
Dim I2 As Long
Dim SW As Long
Dim tmpINDI As tInd

Again:
SW = 0
For I1 = 1 To NofI - 1
    For I2 = I1 + 1 To NofI
        If INDI(I1).Fitness > INDI(I2).Fitness Then
            tmpINDI = INDI(I1)
            INDI(I1) = INDI(I2)
            INDI(I2) = tmpINDI
            SW = SW + 1
        End If
    Next I2
Next I1
'Stop
If SW <> 0 Then GoTo Again
End Sub
Private Sub QuickSortFitness(ByVal lngFirst As Long, ByVal lngLast As Long)
Dim lngLow                  As Long
Dim lngHigh                 As Long
Dim dblMidValue             As Double
Dim HelpChromo              As ChromosomeindividualType
Dim tmpINDI         As tInd
lngLow = lngFirst
lngHigh = lngLast
dblMidValue = Abs(INDI((lngFirst + lngLast) \ 2).Fitness)
Do
    'While Abs(Chromosome(lngLow).Fitness) < dblMidValue
    While INDI(lngLow).Fitness < dblMidValue
        lngLow = lngLow + 1
    Wend
    'While Abs(Chromosome(lngHigh).Fitness) > dblMidValue
    While INDI(lngHigh).Fitness > dblMidValue
        lngHigh = lngHigh - 1
    Wend
    If lngLow <= lngHigh Then
        GoSub swap
        lngLow = lngLow + 1
        lngHigh = lngHigh - 1
End If
Loop While lngLow <= lngHigh
If lngFirst < lngHigh Then QuickSortFitness lngFirst, lngHigh
If lngLow < lngLast Then QuickSortFitness lngLow, lngLast
Exit Sub
swap:
'HelpChromo = Chromosome(lngLow)
'Chromosome(lngLow) = Chromosome(lngHigh)
'Chromosome(lngHigh) = HelpChromo
tmpINDI = INDI(lngLow)
INDI(lngLow) = INDI(lngHigh)
INDI(lngHigh) = tmpINDI
Return
End Sub
Private Function EdgeRecombinationCrossover(P1 As Long, P2 As Long) As tInd
Dim NI As tInd
Dim NeiList() As New Collection
Dim CHILD As New Collection
Dim getCHILD As New Collection
Dim NOG As Long
Dim G As Long
Dim G1 As Long
Dim G2 As Long
Dim V1 As Long
Dim V2 As Long
Dim p1G1 As Long
Dim p1G2 As Long
Dim p2G1 As Long
Dim p2G2 As Long
Dim x As Long
Dim Z As Long
Dim GG As Long
Dim E As Integer
Dim S As String
NI.Fitness = 1E+99
NI.NofG = INDI(P1).NofG
NI.Wheel = 0 'da calcolare
NI.IsChanged = True
ReDim NI.Gene(NI.NofG)
ReDim NeiList(NI.NofG)
NOG = NI.NofG
'Generate Neighbor List
For G = 1 To NOG
    G1 = G - 1
    G2 = G + 1
    If G1 < 1 Then G1 = NOG
    If G2 > NOG Then G2 = 1
    V1 = INDI(P1).Gene(G)
V2 = INDI(P2).Gene(G)
p1G1 = INDI(P1).Gene(G1)
p1G2 = INDI(P1).Gene(G2)
p2G1 = INDI(P2).Gene(G1)
p2G2 = INDI(P2).Gene(G2)
If collNotInColl(NeiList(V1), p1G1) Then NeiList(V1).Add p1G1
If collNotInColl(NeiList(V1), p1G2) Then NeiList(V1).Add p1G2
If collNotInColl(NeiList(V2), p2G1) Then NeiList(V2).Add p2G1
If collNotInColl(NeiList(V2), p2G2) Then NeiList(V2).Add p2G2
getCHILD.Add G
Next
'Stop
x = INDI(P1).Gene(1)
GG = 0
Do
GG = GG + 1: If GG > NOG - 1 Then Exit Do
CHILD.Add x
collRemoveItem getCHILD, x
For G = 1 To NOG
collRemoveItem NeiList(G), x
Next
'if X's neighbor list is empty:
If NeiList(x).Count = 0 Then
    'Z = random node not already in CHILD
    Z = collPickFrom(getCHILD)
    'If Not (collNotInColl(CHILD, Z)) Then Stop
Else
    Dim Min
    Min = 99999999
    For E = 1 To NeiList(x).Count
        If NeiList(NeiList(x).Item(E)).Count < Min Then
            Min = NeiList(NeiList(x).Item(E)).Count
            Z = NeiList(x).Item(E) 'E
        End If
    Next
    Stop
    collRemoveItem getCHILD, Z
End If
Next
x = Z
Loop While True
Z = collPickFrom(getCHILD)
CHILD.Add Z
For G = 1 To NOG
    Debug.Print CHILD.Item(G);
    NI.Gene(G) = CHILD.Item(G)
Next
Set CHILD = Nothing
Set getCHILD = Nothing
For G = 1 To NOG
    Set NeiList(G) = Nothing
Next
EdgeRecombinationCrossover = NI
Exit Function
debugNeilist:
S = ""
For G = 1 To NOG
    S = S + CStr(INDI(P1).Gene(G)) & vbTab
Next: S = S & vbCrLf
For G = 1 To NOG
    S = S + CStr(INDI(P2).Gene(G)) & vbTab
Next: S = S & vbCrLf
Debug.Print S
S = ""
For G1 = 1 To NOG
    S = S & G1 & " |" & vbTab
    For G2 = 1 To NeiList(G1).Count
        S = S & NeiList(G1).Item(G2) & vbTab
    Next: S = S & vbCrLf
Next
Debug.Print S
Return
End Function
Private Function collNotInColl(C As Collection, Val) As Boolean
    Dim E As Integer
    collNotInColl = True
    'If C.Count = 0 Then Stop
    For E = 1 To C.Count
        If C.Item(E) = Val Then collNotInColl = False: Exit For
    Next
End Function
Private Sub collRemoveItem(ByRef C As Collection, Val)
    Dim E As Integer
    If C.Count = 0 Then Exit Sub
    E = 0
    Do
        E = E + 1
        If C.Item(E) = Val Then C.Remove E: Exit Do
    Loop While E < C.Count
End Sub
Private Function collPickFrom(ByRef C As Collection) As Integer
    Dim E As Integer
    'Stop
    'If C.Count = 0 Then Stop
    E = Int(Rnd * C.Count) + 1
    collPickFrom = C.Item(E)
    'C.Remove E
End Function
Private Function collNotInColl(C As Collection, Val) As Boolean
Dim E As Integer
  collNotInColl = True
  If C.Count = 0 Then Stop
  For E = 1 To C.Count
    If C.Item(E) = Val Then collNotInColl = False: Exit For
  Next
End Function
Private Sub collRemoveItem(ByRef C As Collection, Val)
Dim E As Integer
  If C.Count = 0 Then Exit Sub
  E = 0
  Do
    E = E + 1
    If C.Item(E) = Val Then C.Remove E: Exit Do
  Loop While E < C.Count
End Sub
Private Function collPickFrom(ByRef C As Collection) As Integer
Dim E As Integer
  Stop
  If C.Count = 0 Then Stop
  E = Int(Rnd * C.Count) + 1
  collPickFrom = C.Item(E)
  C.Remove E