```
frmCreditCard - 1

Const cards = 3

Dim totpayreverse As Double
Dim totpay As Double
Dim enoughdough As Boolean
Dim pay(1 To cards) As Double
Dim payfirst(1 To 6, 1 To cards) As Integer
Dim permu As Integer

Dim bal(1 To cards) As Double
Dim minrate(1 To cards) As Double
Dim apr(1 To cards) As Double
Dim minfix(1 To cards) As Double
Dim monthpay, monthpaytemp As Double


Private Sub cmdCombinations_Click()

Dim x As Integer
Dim totalcost(1 To 6) As Double
Dim min, maxapr As Double
Dim minindex, maxaprindex As Integer
Dim nbal1, nbal2, nbal3 As Double
Dim napr1, napr2, napr3
Dim nmonthpay As Double
Dim nminp, totnminp As Double

Call loadvariables 'to load payfirst array

nbal1 = 2000
minfix(1) = 15
minfix(2) = 20
minfix(3) = 10



  For nbal2 = 107 To 10007 Step 100
    For napr2 = 0.19 To 0.19 Step -0.01
     For nbal3 = 702 To 702 Step -100
      For napr3 = 0.08 To 0.25 Step 0.01
       For nmonthpay = 1003 To 53 Step -50
        For napr1 = 0.14 To 0.14 Step 0.01
        For nminrate1 = 0.02 To 0.04 Step 0.01
         For nminrate2 = 0.04 To 0.02 Step -0.01
          For nminrate3 = 0.02 To 0.04 Step 0.01


            'calculate total minimum payment and store in totnminp
             totnminp = 0
             bal(1) = nbal1
             bal(2) = nbal2
             bal(3) = nbal3
             minrate(1) = nminrate1
             minrate(2) = nminrate2
             minrate(3) = nminrate3

             For x = 1 To cards
                nminp = minrate(x) * bal(x)
                If nminp <= minfix(x) Then nminp = minfix(x)
                If bal(x) <= nminp Then nminp = bal(x)
                totnminp = totnminp + nminp
             Next x

      'no aprs can be equal and must have enough dough to cover min payments
       If (napr3 <> napr2) And (napr3 <> napr1) And (napr1 <> napr2) And (totnminp <= nmonthpay) Th
en

         'find maximum apr card
```

```
        apr(1) = napr1
        apr(2) = napr2
        apr(3) = napr3
        maxaprindex = cards
        maxapr = apr(cards)
        For x = 1 To cards
          If apr(x) > maxapr Then
            maxapr = apr(x)
            maxaprindex = x
          End If
        Next x

        'do computation
       For permu = 1 To 6

         'Manually load variables
         bal(1) = nbal1
         bal(2) = nbal2
         bal(3) = nbal3
         apr(1) = napr1
         apr(2) = napr2
         apr(3) = napr3

         monthpay = nmonthpay

         Call Calculate
         totalcost(permu) = totpay
        Next permu

         'find minimum total cost
         minindex = 6
         min = totalcost(6)
         For x = 1 To 6
           If totalcost(x) < min Then
             min = totalcost(x)
             minindex = x
           End If
         Next x

         'compare - if first to pay is NOT one with highest apr and costs are less then
         If (payfirst(maxaprindex, 1) <> minindex) And _
          (totalcost(payfirst(maxaprindex, 1)) < (totalcost(minindex) - 0.01)) Then

           listStopBox.AddItem "bal 1= " & nbal1 & "bal 2= " & nbal2 & " bal 3= " & nbal3
           listStopBox.AddItem "apr 1= " & apr(1) & "apr 2= " & apr(2) & " apr 3= " & apr(3)
           listStopBox.AddItem "min1= " & minrate(1) & "min2= " & minrate(2) & " min3= " & minrate
(3)
           listStopBox.AddItem "monthpay= " & nmonthpay

           listStopBox.AddItem "123 = " & totalcost(1)
           listStopBox.AddItem "321 = " & totalcost(2)
           listStopBox.AddItem "213 = " & totalcost(3)
           listStopBox.AddItem "231 = " & totalcost(4)
           listStopBox.AddItem "132 = " & totalcost(5)
           listStopBox.AddItem "312 = " & totalcost(6)
           listStopBox.AddItem "payfirst = " & payfirst(maxaprindex, 1) & " " & minindex
           listStopBox.AddItem totalcost(payfirst(maxaprindex, 1)) & " " & totalcost(minindex) + 0
.01
           Stop
         End If



        End If 'apr 3
          Next nminrate3
          Next nminrate2
          Next nminrate1
          Next napr1
```

```
        Next nmonthpay
      Next napr3
    Next nbal3
  Next napr2
Next nbal2



End Sub

Private Sub cmdFindLowest_Click()

Dim x As Integer
Dim totalcost(1 To 6) As Double
Dim min As Double
Dim minindex As Integer


For permu = 1 To 6
    Call loadvariables
    Call Calculate
    totalcost(permu) = totpay
Next permu

minindex = 6
min = totalcost(6)
For x = 1 To 6
   If totalcost(x) < min Then
       min = totalcost(x)
       minindex = x
   End If
Next x

'Display everything again for convenience
permu = minindex
Call loadvariables
Call Calculate
listOutput.AddItem "123 = " & totalcost(1)
listOutput.AddItem "321 = " & totalcost(2)
listOutput.AddItem "213 = " & totalcost(3)
listOutput.AddItem "231 = " & totalcost(4)
listOutput.AddItem "132 = " & totalcost(5)
listOutput.AddItem "312 = " & totalcost(6)


listOutput.AddItem "The best method is:"
For x = 1 To cards
  listOutput.AddItem payfirst(minindex, x)
Next x

listOutput.AddItem "The smallest total is: " & min
listOutput.AddItem "Diff from 123 is: " & (totalcost(1) - min)

End Sub

Private Sub cmdPay2_Click()

    Call loadvariables
    permu = 2
    Call Calculate

   listOutput.AddItem "total paid (cards, total):"

   listOutput.AddItem Format(pay(1), "####0.00") & Space(2) _
                      & Format(pay(2), "####0.00") & Space(2) _
                      & Format(pay(3), "####0.00") & Space(2) _
                      & Format(totpay, "####0.00")
```

```
End Sub


Private Sub cmdPay1_Click()

    Call loadvariables
    permu = 1
    Call Calculate

   listOutput.AddItem "total paid (cards, total):"

   listOutput.AddItem Format(pay(1), "####0.00") & Space(2) _
                      & Format(pay(2), "####0.00") & Space(2) _
                      & Format(pay(3), "####0.00") & Space(2) _
                      & Format(totpay, "####0.00")


End Sub




Private Sub Calculate()


    Dim prevbal(1 To cards) As Double
    Dim minp(1 To cards) As Double
    Dim x, y As Integer
    Dim stayin As Boolean
    Dim OuputString As String
    Dim mintotal As Double


    For x = 1 To cards
        pay(x) = 0
    Next x

    listOutput.Clear
    listOutput.AddItem "Balance Payment, Balance Payment, etc."
    enoughdough = True

    mintotal = 0
    For x = 1 To cards
    mintotal = mintotal + minrate(x) * bal(x)
    Next x

    If mintotal > monthpay Then
      listOutput.AddItem "Not enough dough"
      enoughdough = False
    End If


    Do
            stayin = False
            monthpaytemp = monthpay

             'calculate minimum payment and store in minp(x)
             For x = 1 To cards
                minp(x) = minrate(x) * bal(x)
                If minp(x) <= minfix(x) Then minp(x) = minfix(x)
                If bal(x) <= minp(x) Then minp(x) = bal(x)

                'store untouched balance for output
```

```
            prevbal(x) = bal(x)
         Next x


         'pay minimums on all cards and compute left over in monthpaytemp
         For x = 1 To cards
            bal(x) = bal(x) - minp(x)
            monthpaytemp = monthpaytemp - minp(x)
            pay(x) = pay(x) + minp(x)
         Next x

        'Disperse left over according to pay plan until monthpaytemp is depleted
        'or all balances or zero- payfirst(y) stores sequence of cards.. (1,3,2), etc.

        Do
          For y = 1 To cards


            'balance is bigger
             If bal(payfirst(permu, y)) >= monthpaytemp Then
                bal(payfirst(permu, y)) = bal(payfirst(permu, y)) - monthpaytemp
                pay(payfirst(permu, y)) = pay(payfirst(permu, y)) + monthpaytemp
                minp(payfirst(permu, y)) = minp(payfirst(permu, y)) + monthpaytemp
                monthpaytemp = 0
             End If

             'balance is smaller
             If bal(payfirst(permu, y)) > 0 And bal(payfirst(permu, y)) < monthpaytemp Then
                monthpaytemp = monthpaytemp - bal(payfirst(permu, y))
                pay(payfirst(permu, y)) = pay(payfirst(permu, y)) + bal(payfirst(permu, y))
                minp(payfirst(permu, y)) = minp(payfirst(permu, y)) + bal(payfirst(permu, y))
                bal(payfirst(permu, y)) = 0
             End If



          Next y


             'set exit flag if all balances are zero
             For x = 1 To cards
                If bal(x) > 0 Then stayin = True
             Next x


         Loop Until (monthpaytemp = 0 Or stayin = False)


         'build output string and calculate interest and add to balance
         For x = 1 To cards
            OutputString = OutputString & Format(prevbal(x), "#####0.00") & Space(2)
            OutputString = OutputString & Format(minp(x), "#####0.00") & Space(2)
            bal(x) = bal(x) + (bal(x) * (apr(x) / 12))
         Next x

      listOutput.AddItem OutputString
      OutputString = ""

   Loop Until stayin = False

   totpay = 0
   For x = 1 To cards
   totpay = totpay + pay(x)
   Next x

End Sub

Sub loadvariables()
```

```
payfirst(1, 1) = 1: payfirst(1, 2) = 2: payfirst(1, 3) = 3
payfirst(2, 1) = 3: payfirst(2, 2) = 2: payfirst(2, 3) = 1
payfirst(3, 1) = 2: payfirst(3, 2) = 1: payfirst(3, 3) = 3
payfirst(4, 1) = 2: payfirst(4, 2) = 3: payfirst(4, 3) = 1
payfirst(5, 1) = 1: payfirst(5, 2) = 3: payfirst(5, 3) = 2
payfirst(6, 1) = 3: payfirst(6, 2) = 1: payfirst(6, 3) = 2

bal(1) = txtBalance1.Text
bal(2) = txtBalance2.Text
bal(3) = txtBalance3.Text
minrate(1) = txtMinrate1.Text
minrate(2) = txtMinrate2.Text
minrate(3) = txtMinrate3.Text
apr(1) = txtInterest1.Text
apr(2) = txtInterest2.Text
apr(3) = txtInterest3.Text
minfix(1) = txtMinfloor1.Text
minfix(2) = txtMinfloor2.Text
minfix(3) = txtMinfloor3.Text
monthpay = txtMonthpay.Text

End Sub
```