```
'This is the code to generate the data presented in Figure 3 of the the paper
'"Induction with and without natural properties: a new approach to the New Riddle of Induction".
'The program was created in Visual Studio 2019.
'To run the program, create a windows form application with single button "Button 1", paste this code into Form1.vb, and place a breakpoint at line 502 (End Sub).
'After executing the program, press Button 1.
'Sample_Size is currently set to 8. In order to generate the data for Figure 3, you will need to run the program 6 times for the 6 sample sizes presented in the figure.
'Once the program breaks at line 502, data of the type presented in Figure 3 will have been outputted to an Excel file.

Imports System.Math


Public Class Form1

    Dim Big_Loop_Size As Integer = 1000
    Dim Number_of_Objects As Integer = 1000000
    Dim Sample_Size As Double = 8
    Dim Number_of_Dimensions As Integer = 4
    Dim Number_of_Natural_Categories As Double = 8
    Dim Std_Dev_for_Natural_Categories As Double = 0.2
    Dim Number_of_Centroids As Integer = 8 'Number_of_Natural_Categories

    Dim Natural_Category_Means(Number_of_Natural_Categories, Number_of_Dimensions) As Double
    Dim Natural_Category_Probabilities(Number_of_Natural_Categories) As Double

    Dim Natural_Category_Counts(Number_of_Natural_Categories) As Double

    Dim Object_Coordinates(Number_of_Objects, Number_of_Dimensions) As Double

    Dim Centroid_Coordinates(Number_of_Centroids, Number_of_Dimensions) As Double
    Dim Provisional_Centroid_Coordinates(Number_of_Centroids, Number_of_Dimensions) As Double
    Dim Top_Centroid_Coordinates(Number_of_Centroids, Number_of_Dimensions) As Double

    Dim Distance_Holder1 As Double
    Dim Distance_Holder2 As Double

    Dim Object_Centroid_Assigment(Number_of_Objects) As Double
    Dim Object_Provisional_Centroid_Assigment(Number_of_Objects) As Double
    Dim Object_Updated_Centroid_Assigment(Number_of_Objects) As Double
    Dim Object_Top_Centroid_Assigment(Number_of_Objects) As Double

    Dim Updated_Centroid_Coordinates(Number_of_Centroids, Number_of_Dimensions) As Double

    Dim Total_Object_to_Assigned_Top_Centroids_Distance As Double
    Dim Total_Object_to_Assigned_Provisional_Centroids_Distance As Double
    Dim Total_Object_to_Assigned_Updated_Centroids_Distance As Double

    Dim Number_of_Objects_Attached_to_Centroid_Counter As Double

    Dim objectcounter As Integer

    Dim Sample_Object_Coordinates(Sample_Size, Number_of_Dimensions) As Double
    Dim Sample_Object_Top_Centroid_Assigment(Number_of_Objects) As Double

    Dim Object_to_Fitted_Category_Count(Number_of_Centroids) As Double
    Dim Object_to_Fitted_Category_in_Sample_Count(Number_of_Centroids) As Double

    Dim Mean_Distance_from_frequencies_in_Universe_and_Sample_for_fitted_Classes As Double

    Dim Fitted_Class_Frequencies_in_Sample(Number_of_Centroids) As Double
    Dim Fitted_Class_Frequencies_in_Universe(Number_of_Centroids) As Double
    Dim Rectangular_Category_Frequencies_in_Sample(Number_of_Centroids) As Double
    Dim Rectangular_Category_Frequencies_in_Universe(Number_of_Centroids) As Double
    Dim Non_convex_Category_Frequencies_in_Sample(Number_of_Centroids) As Double
    Dim Non_convex_Category_Frequencies_in_Universe(Number_of_Centroids) As Double

    Dim Fitted_Category_Accuracy(Big_Loop_Size) As Double
    Dim Rectangular_Category_Accuracy(Big_Loop_Size) As Double
    'Dim Non_convex_Category_Accuracy(Big_Loop_Size) As Double

    Dim Mean_Fitted_Category_Accuracy As Double
    Dim Mean_Rectangular_Category_Accuracy As Double
    'Dim Mean_Non_convex_Category_Accuracy As Double

    Dim RandomClass As New Random()
    Dim RandomNumber As Double

    Dim holder As Double

    Dim objApp As Microsoft.Office.Interop.Excel.Application 'used in accessing an manipulating an excel spreadsheet
    Dim objBook As Microsoft.Office.Interop.Excel._Workbook 'for excel



   Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click

        Dim objBooks As Microsoft.Office.Interop.Excel.Workbooks 'for excel
        Dim objSheets As Microsoft.Office.Interop.Excel.Sheets 'for excel
        Dim objSheet As Microsoft.Office.Interop.Excel._Worksheet 'for excel
        Dim range As Microsoft.Office.Interop.Excel.Range 'for excel

        ' Create a new instance of Excel and start a new workbook:
        objApp = New Microsoft.Office.Interop.Excel.Application()
        objBooks = objApp.Workbooks
        objBook = objBooks.Add
        objSheets = objBook.Worksheets
        objSheet = objSheets(1)

        'range = objSheet.Range("A1", Reflection.Missing.Value) 'for excel
        'range = range.Resize(Players, Rounds + 1) 'for excel

        'Dim saRet(Players, Rounds + 1) As Double 'This matrix will temporarily store the match outcomes and the predictions of the nonMIs.

        'range.Value = saRet 'indicates the cells of the spreadsheet that will be loaded into saRet

        'Return control of Excel to the user.
        objApp.Visible = True
        objApp.UserControl = True

        'Clean up a little.
        range = Nothing
        objSheet = Nothing
        objSheets = Nothing
        objBooks = Nothing

        Randomize()




        For big_loop = 1 To Big_Loop_Size


            For x = 1 To Number_of_Natural_Categories
                Natural_Category_Counts(x) = 0
            Next


            'Choose the mean values for categories
            For x = 1 To Number_of_Natural_Categories
                For y = 1 To Number_of_Dimensions
                    RandomNumber = RandomClass.NextDouble()
                    Natural_Category_Means(x, y) = RandomNumber
                Next
            Next


            'Determine Number of Objects for each Category:
            holder = 0
            For x = 1 To Number_of_Natural_Categories
                RandomNumber = RandomClass.NextDouble()
                Natural_Category_Probabilities(x) = RandomNumber
                holder = holder + RandomNumber
            Next

            For x = 1 To Number_of_Natural_Categories
                Natural_Category_Probabilities(x) = Natural_Category_Probabilities(x) / holder
            Next
```

```
        For x = 1 To Number_of_Objects
            RandomNumber = RandomClass.NextDouble()
            'RandomNumber = RandomNumber * Number_of_Natural_Categories
            holder = 0
            For y = 1 To Number_of_Natural_Categories
                holder = holder + Natural_Category_Probabilities(y)
                If RandomNumber < holder Then
                    Natural_Category_Counts(y) = Natural_Category_Counts(y) + 1
                    y = Number_of_Natural_Categories
                End If
            Next
        Next


        'Determine the position of each object:
        objectcounter = 0

        holder = Math.Ceiling(Number_of_Dimensions / 2) * 2

        For x = 1 To Number_of_Natural_Categories

            For y = 1 To Natural_Category_Counts(x)

                objectcounter = objectcounter + 1

                GaussNumDist(0, Std_Dev_for_Natural_Categories, holder)

                For z = 1 To Number_of_Dimensions

                    Object_Coordinates(objectcounter, z) = Natural_Category_Means(x, z) + GaussNumArray(z)

                    If Object_Coordinates(objectcounter, z) > 1 Then
                        Object_Coordinates(objectcounter, z) = 1
                    End If
                    If Object_Coordinates(objectcounter, z) < 0 Then
                        Object_Coordinates(objectcounter, z) = 0
                    End If

                Next

            Next

        Next


        'Generate a sample:
        For x = 1 To Sample_Size
            RandomNumber = RandomClass.NextDouble()
            RandomNumber = RandomNumber * Number_of_Objects
            RandomNumber = Math.Ceiling(RandomNumber)
            For y = 1 To Number_of_Dimensions
                Sample_Object_Coordinates(x, y) = Object_Coordinates(RandomNumber, y)
            Next

        Next


        'loop to run for different choices of initial centroids:
        Total_Object_to_Assigned_Top_Centroids_Distance = 100000000
        For L = 1 To 10


            'Choose initial centroids:
            For x = 1 To Number_of_Centroids
                For y = 1 To Number_of_Dimensions
                    RandomNumber = RandomClass.NextDouble()
                    Provisional_Centroid_Coordinates(x, y) = RandomNumber
                Next
            Next

            'Assign Objects to Centroids:
            For a = 1 To Sample_Size
                Distance_Holder2 = 10
                For x = 1 To Number_of_Centroids
                    Distance_Holder1 = 0
                    For y = 1 To Number_of_Dimensions
                        Distance_Holder1 = Distance_Holder1 + distance(Provisional_Centroid_Coordinates(x, y), Sample_Object_Coordinates(a, y))
                    Next
                    If Distance_Holder1 < Distance_Holder2 Then
                        Object_Provisional_Centroid_Assigment(a) = x
                        Distance_Holder2 = Distance_Holder1
                    End If
                Next
            Next


            'Compute Total Subclass to Assigned Centroids Distance:
            Total_Object_to_Assigned_Provisional_Centroids_Distance = 0
            For x = 1 To Number_of_Centroids
                For a = 1 To Sample_Size
                    If Object_Provisional_Centroid_Assigment(a) = x Then
                        For y = 1 To Number_of_Dimensions
                            Total_Object_to_Assigned_Provisional_Centroids_Distance = Total_Object_to_Assigned_Provisional_Centroids_Distance + distance(Provisional_Centroid_Coordinates(x, y),
Sample_Object_Coordinates(a, y))
                        Next
                    End If
                Next
            Next


            'Iterate Lloyd's Algorythm:
            For r = 1 To 10

                'Update Centroids:
                For x = 1 To Number_of_Centroids
                    Number_of_Objects_Attached_to_Centroid_Counter = 0
                    For y = 1 To Number_of_Dimensions
                        Updated_Centroid_Coordinates(x, y) = 0
                    Next
                    For a = 1 To Sample_Size
                        If Object_Provisional_Centroid_Assigment(a) = x Then
                            For y = 1 To Number_of_Dimensions
                                Updated_Centroid_Coordinates(x, y) = Updated_Centroid_Coordinates(x, y) + Sample_Object_Coordinates(a, y)
                            Next
                            Number_of_Objects_Attached_to_Centroid_Counter = Number_of_Objects_Attached_to_Centroid_Counter + 1
                        End If
                    Next
                    If Number_of_Objects_Attached_to_Centroid_Counter > 0 Then
                        For y = 1 To Number_of_Dimensions
                            Updated_Centroid_Coordinates(x, y) = Updated_Centroid_Coordinates(x, y) / Number_of_Objects_Attached_to_Centroid_Counter
                        Next
                    Else
                        For y = 1 To Number_of_Dimensions
                            RandomNumber = RandomClass.NextDouble()
                            Updated_Centroid_Coordinates(x, y) = RandomNumber
                        Next
                    End If
                Next

                'Assign Objects to Centroids:
                For a = 1 To Sample_Size
                    Distance_Holder2 = 10
                    For x = 1 To Number_of_Centroids
                        Distance_Holder1 = 0
                        For y = 1 To Number_of_Dimensions
                            Distance_Holder1 = Distance_Holder1 + distance(Updated_Centroid_Coordinates(x, y), Sample_Object_Coordinates(a, y))
                        Next
                        If Distance_Holder1 < Distance_Holder2 Then
                            Object_Updated_Centroid_Assigment(a) = x
                            Distance_Holder2 = Distance_Holder1
                        End If
                    Next
                Next


                'Compute Total Object to Assigned Centroids Distance:
                Total_Object_to_Assigned_Updated_Centroids_Distance = 0
                For x = 1 To Number_of_Centroids
                    For a = 1 To Sample_Size
                        If Object_Updated_Centroid_Assigment(a) = x Then
```

```
                            For y = 1 To Number_of_Dimensions
                                Total_Object_to_Assigned_Updated_Centroids_Distance = Total_Object_to_Assigned_Updated_Centroids_Distance + distance(Updated_Centroid_Coordinates(x, y),
Sample_Object_Coordinates(a, y))
                            Next
                        End If
                    Next
                Next


                'Test whether Updated Centroids are better.
                If Total_Object_to_Assigned_Updated_Centroids_Distance < Total_Object_to_Assigned_Provisional_Centroids_Distance Then
                    Total_Object_to_Assigned_Provisional_Centroids_Distance = Total_Object_to_Assigned_Updated_Centroids_Distance
                    For x = 1 To Number_of_Centroids
                        For y = 1 To Number_of_Dimensions
                            Provisional_Centroid_Coordinates(x, y) = Updated_Centroid_Coordinates(x, y)
                        Next
                    Next
                    For a = 1 To Sample_Size
                        Object_Provisional_Centroid_Assigment(a) = Object_Updated_Centroid_Assigment(a)
                    Next
                Else
                    r = 1000
                End If

            Next



            If Total_Object_to_Assigned_Provisional_Centroids_Distance < Total_Object_to_Assigned_Top_Centroids_Distance Then
                Total_Object_to_Assigned_Top_Centroids_Distance = Total_Object_to_Assigned_Provisional_Centroids_Distance
                For x = 1 To Number_of_Centroids
                    For y = 1 To Number_of_Dimensions
                        Top_Centroid_Coordinates(x, y) = Provisional_Centroid_Coordinates(x, y)
                    Next
                Next
                For a = 1 To Sample_Size
                    Sample_Object_Top_Centroid_Assigment(a) = Object_Provisional_Centroid_Assigment(a)
                Next
            End If


        Next


        'compute frequencies of fitted class membership in sample

        For x = 1 To Number_of_Centroids
            Fitted_Class_Frequencies_in_Sample(x) = 0
        Next
        For x = 1 To Sample_Size
            For y = 1 To Number_of_Centroids
                If Sample_Object_Top_Centroid_Assigment(x) = y Then
                    Fitted_Class_Frequencies_in_Sample(y) = Fitted_Class_Frequencies_in_Sample(y) + 1
                End If
            Next
        Next
        For y = 1 To Number_of_Centroids
            Fitted_Class_Frequencies_in_Sample(y) = Fitted_Class_Frequencies_in_Sample(y) / Sample_Size
        Next

        'compute frequencies of fitted class membership in population
        'begin by assigning objects to centroids:
        For a = 1 To Number_of_Objects
            Distance_Holder2 = 10000
            For x = 1 To Number_of_Centroids
                Distance_Holder1 = 0
                For y = 1 To Number_of_Dimensions
                    Distance_Holder1 = Distance_Holder1 + distance(Top_Centroid_Coordinates(x, y), Object_Coordinates(a, y))
                Next
                If Distance_Holder1 < Distance_Holder2 Then
                    Object_Updated_Centroid_Assigment(a) = x
                    Distance_Holder2 = Distance_Holder1
                End If
            Next
        Next

        For x = 1 To Number_of_Centroids
            Fitted_Class_Frequencies_in_Universe(x) = 0
        Next
        For x = 1 To Number_of_Objects
            For y = 1 To Number_of_Centroids
                If Object_Updated_Centroid_Assigment(x) = y Then
                    Fitted_Class_Frequencies_in_Universe(y) = Fitted_Class_Frequencies_in_Universe(y) + 1
                End If
            Next
        Next
        For y = 1 To Number_of_Centroids
            Fitted_Class_Frequencies_in_Universe(y) = Fitted_Class_Frequencies_in_Universe(y) / Number_of_Objects
        Next


        'Now compute the sample and population frequencies for rectanglular categories:
        For x = 1 To Number_of_Centroids
            Rectangular_Category_Frequencies_in_Sample(x) = 0
        Next
        For x = 1 To Sample_Size
            For y = 1 To Number_of_Centroids
                If (y - 1) / Number_of_Centroids <= Sample_Object_Coordinates(x, 1) And Sample_Object_Coordinates(x, 1) <= y / Number_of_Centroids Then
                    Rectangular_Category_Frequencies_in_Sample(y) = Rectangular_Category_Frequencies_in_Sample(y) + 1
                End If
            Next
        Next
        For x = 1 To Number_of_Centroids
            Rectangular_Category_Frequencies_in_Sample(x) = Rectangular_Category_Frequencies_in_Sample(x) / Sample_Size
        Next

        For x = 1 To Number_of_Centroids
            Rectangular_Category_Frequencies_in_Universe(x) = 0
        Next
        For x = 1 To Number_of_Objects
            For y = 1 To Number_of_Centroids
                If (y - 1) / Number_of_Centroids <= Object_Coordinates(x, 1) And Object_Coordinates(x, 1) <= y / Number_of_Centroids Then
                    Rectangular_Category_Frequencies_in_Universe(y) = Rectangular_Category_Frequencies_in_Universe(y) + 1
                End If
            Next
        Next
        For x = 1 To Number_of_Centroids
            Rectangular_Category_Frequencies_in_Universe(x) = Rectangular_Category_Frequencies_in_Universe(x) / Number_of_Objects
        Next




        ''Now compute the sample and population frequencies for non-convex categories that are unions of rectangular categories:
        'For x = 1 To Number_of_Centroids
        '    Non_convex_Category_Frequencies_in_Sample(x) = 0
        'Next
        'For x = 1 To Sample_Size
        '    For y = 1 To Number_of_Centroids
        '        If (((y - 1) / (Number_of_Centroids * 2)) + 0.5 <= Sample_Object_Coordinates(x, 1) And Sample_Object_Coordinates(x, 1) <= (y / (Number_of_Centroids * 2)) + 0.5) Or ((y - 1) /
(Number_of_Centroids * 2) <= Sample_Object_Coordinates(x, 1) And Sample_Object_Coordinates(x, 1) <= y / (Number_of_Centroids * 2)) Then
        '            Non_convex_Category_Frequencies_in_Sample(y) = Non_convex_Category_Frequencies_in_Sample(y) + 1
        '        End If
        '    Next
        'Next
        'For x = 1 To Number_of_Centroids
        '    Non_convex_Category_Frequencies_in_Sample(x) = Non_convex_Category_Frequencies_in_Sample(x) / Sample_Size
        'Next

        'For x = 1 To Number_of_Centroids
        '    Non_convex_Category_Frequencies_in_Universe(x) = 0
        'Next
        'For x = 1 To Number_of_Objects
        '    For y = 1 To Number_of_Centroids
        '        If (((y - 1) / (Number_of_Centroids * 2)) + 0.5 <= Object_Coordinates(x, 1) And Object_Coordinates(x, 1) <= (y / (Number_of_Centroids * 2)) + 0.5) Or ((y - 1) / (Number_of_Centroids *
2) <= Object_Coordinates(x, 1) And Object_Coordinates(x, 1) <= y / (Number_of_Centroids * 2)) Then
        '            Non_convex_Category_Frequencies_in_Universe(y) = Non_convex_Category_Frequencies_in_Universe(y) + 1
        '        End If
        '    Next
        'Next
```

```
            'For x = 1 To Number_of_Centroids
            '    Non_convex_Category_Frequencies_in_Universe(x) = Non_convex_Category_Frequencies_in_Universe(x) / Number_of_Objects
            'Next


            For x = 1 To Number_of_Centroids
                Fitted_Category_Accuracy(big_loop) = Fitted_Category_Accuracy(big_loop) + Math.Abs(Fitted_Class_Frequencies_in_Sample(x) - Fitted_Class_Frequencies_in_Universe(x))
                Rectangular_Category_Accuracy(big_loop) = Rectangular_Category_Accuracy(big_loop) + Math.Abs(Rectangular_Category_Frequencies_in_Sample(x) - Rectangular_Category_Frequencies_in_Universe(x))
                'Non_convex_Category_Accuracy(big_loop) = Non_convex_Category_Accuracy(big_loop) + Math.Abs(Non_convex_Category_Frequencies_in_Sample(x) - Non_convex_Category_Frequencies_in_Universe(x))
            Next

            Fitted_Category_Accuracy(big_loop) = Fitted_Category_Accuracy(big_loop) / Number_of_Centroids
            Rectangular_Category_Accuracy(big_loop) = Rectangular_Category_Accuracy(big_loop) / Number_of_Centroids
            'Non_convex_Category_Accuracy(big_loop) = Non_convex_Category_Accuracy(big_loop) / Number_of_Centroids


        Next


        For x = 1 To Big_Loop_Size
            Mean_Fitted_Category_Accuracy = Mean_Fitted_Category_Accuracy + Fitted_Category_Accuracy(x)
            Mean_Rectangular_Category_Accuracy = Mean_Rectangular_Category_Accuracy + Rectangular_Category_Accuracy(x)
            'Mean_Non_convex_Category_Accuracy = Mean_Non_convex_Category_Accuracy + Non_convex_Category_Accuracy(x)
        Next


        Mean_Fitted_Category_Accuracy = Mean_Fitted_Category_Accuracy / Big_Loop_Size
        Mean_Rectangular_Category_Accuracy = Mean_Rectangular_Category_Accuracy / Big_Loop_Size
        'Mean_Non_convex_Category_Accuracy = Mean_Non_convex_Category_Accuracy / Big_Loop_Size



        objApp.Cells(1, 1) = "Big_Loop_Size"
        objApp.Cells(1, 2) = Big_Loop_Size

        objApp.Cells(2, 1) = "Number_of_Objects"
        objApp.Cells(2, 2) = Number_of_Objects

        objApp.Cells(3, 1) = "Sample_Size"
        objApp.Cells(3, 2) = Sample_Size

        objApp.Cells(4, 1) = "Number_of_Dimensions"
        objApp.Cells(4, 2) = Number_of_Dimensions

        objApp.Cells(5, 1) = "Number_of_Natural_Categories"
        objApp.Cells(5, 2) = Number_of_Natural_Categories

        objApp.Cells(6, 1) = "Std_Dev_for_Natural_Categoriese"
        objApp.Cells(6, 2) = Std_Dev_for_Natural_Categories

        objApp.Cells(7, 1) = "Number_of_Centroids"
        objApp.Cells(7, 2) = Number_of_Centroids

        objApp.Cells(8, 1) = "Mean Error Rate for Natural Categories"
        objApp.Cells(8, 2) = Mean_Fitted_Category_Accuracy

        objApp.Cells(9, 1) = "Mean Error Rate for Non-Natural Categories"
        objApp.Cells(9, 2) = Mean_Rectangular_Category_Accuracy

        'objApp.Cells(10, 1) = "Mean_Non_convex_Category_Accuracy"
        'objApp.Cells(10, 2) = Mean_Non_convex_Category_Accuracy


    End Sub




    Public Function distance(ByRef r As Double, ByVal s As Double) As Double
        Dim dist As Double = 0
        dist = (r - s) * (r - s) 'squared
        'dist = Math.Abs(r - s) 'manhatten
        'If dist < 0 Or dist > 10 Then
        '    dist = dist
        'End If
        Return dist
    End Function


End Class


Module Module1
    Friend GaussNumArray() As Double
    Friend intICell As Long

    Friend Function GaussNumDist(ByVal Mean As Double, ByVal StdDev As Double, ByVal SampleSize As Integer)
        intICell = 1                'Loop variable

        ReDim GaussNumArray(SampleSize)

        Do While (intICell < (SampleSize + 1))
            Call NumDist(Mean, StdDev)
            Application.DoEvents()
        Loop
    End Function

    Sub NumDist(ByVal meanin As Double, ByVal sdin As Double)

        'Defining variables
        Dim dblR1 As Double
        Dim dblR2 As Double
        Dim mean As Double
        Dim var As Double
        Dim circ As Double
        Dim trans As Double
        Dim dblY1 As Double
        Dim dblY2 As Double
        Dim Pi As Double
        Pi = 4 * Atan(1)

        'Get two random numbers
        dblR1 = (2 * UniformRandomNumber()) - 1
        dblR2 = (2 * UniformRandomNumber()) - 1

        circ = (dblR1 ^ 2) + (dblR2 ^ 2)        'Radius of circle

        If circ >= 1 Then       'If outside unit circle, then reject number
            Call NumDist(meanin, sdin)
            Exit Sub
        End If

        'Transform to Gaussian
        trans = Sqrt(-2 * Log(circ) / circ)

        dblY1 = (trans * dblR1 * sdin) + meanin
        dblY2 = (trans * dblR2 * sdin) + meanin

        GaussNumArray(intICell) = dblY1   'First number

        'Increase intICell for next random number
        intICell = (intICell + 1)

        GaussNumArray(intICell) = dblY2   'Second number

        'Increase intICell again ready for next call of ConvertNumberDistribution
        intICell = (intICell + 1)

    End Sub

    Friend Function UniformRandomNumber() As Double

        'Defining constants
        Const IM1 As Double = 2147483563
        Const IM2 As Double = 2147483399
        Const AM As Double = (1.0# / IM1)
        Const IMM1 As Double = (IM1 - 1.0#)
        Const IA1 As Double = 40014
        Const IA2 As Double = 40692
```

```
        Const IQ1 As Double = 53668
        Const IQ2 As Double = 52774
        Const IR1 As Double = 12211
        Const IR2 As Double = 3791
        Const NTAB As Double = 32
        Const NDIV As Double = (1.0# + IM1 / NTAB)
        Const ESP As Double = 0.00000012
        Const RNMX As Double = (1.0# - ESP)

        Dim iCell As Integer
        Dim idum As Double
        Dim j As Integer
        Dim k As Long
        Dim temp As Double

        Static idum2 As Long
        Static iy As Long
        Static iv(NTAB) As Long

        idum2 = 123456789
        iy = 0

        'Seed value required is a negative integer (idum)
        Randomize()
        idum = (-Rnd() * 1000)

        'For loop to generate a sequence of random numbers based on idum
        For iCell = 1 To 10
            'Initialize generator
            If (idum <= 0) Then
                'Prevent idum = 0
                If (-(idum) < 1) Then
                    idum = 1
                Else
                    idum = -(idum)
                End If
                idum2 = idum
                For j = (NTAB + 7) To 0
                    k = ((idum) / IQ1)
                    idum = ((IA1 * (idum - (k * IQ1))) - (k * IR1))
                    If (idum < 0) Then
                        idum = (idum + IM1)
                    End If
                    If (j < NTAB) Then
                        iv(j) = idum
                    End If
                Next j
                iy = iv(0)
            End If

            'Start here when not initializing
            k = (idum / IQ1)
            idum = ((IA1 * (idum - (k * IQ1))) - (k * IR1))
            If (idum < 0) Then
                idum = (idum + IM1)
            End If
            k = (idum2 / IQ2)
            idum2 = ((IA2 * (idum2 - (k * IQ2))) - (k * IR2))
            If (idum2 < 0) Then
                idum2 = idum2 + IM2
            End If
            j = (iy / NDIV)
            iy = (iv(j) - idum2)
            iv(j) = idum
            If (iy < 1) Then
                iy = (iy + IMM1)
            End If
            temp = AM * iy
            If (temp <= RNMX) Then
                'Return the value of the random number
                UniformRandomNumber = temp
            End If
        Next iCell
    End Function

End Module
```