```
'This random number generator is a VBA translation of the C program found in
'Law, A. M. and Kelton, W. D., ``Simulation Modeling and Analysis'',
'Singapore: The McGraw-Hill Book Co, pp. 430--431.

Option Explicit

Const MODLUS = 2147483647
Const MULT1 = 24112
Const MULT2 = 26143

'Define Static variable
Dim zrng() As Long

Public Static Sub InitializeRNSeed()

    ReDim zrng(1 To 100) As Long

    zrng(1) = 1973272912
    zrng(2) = 281629770
    zrng(3) = 20006270
    zrng(4) = 1280689831
    zrng(5) = 2096730329
    zrng(6) = 1933576050
    zrng(7) = 913566091
    zrng(8) = 246780520
    zrng(9) = 1363774876
    zrng(10) = 604901985
    zrng(11) = 1511192140
    zrng(12) = 1259851944
    zrng(13) = 824064364
    zrng(14) = 150493284
    zrng(15) = 242708531
    zrng(16) = 75253171
    zrng(17) = 1964472944
    zrng(18) = 1202299975
    zrng(19) = 233217322
    zrng(20) = 1911216000
    zrng(21) = 726370533
    zrng(22) = 403498145
    zrng(23) = 993232223
    zrng(24) = 1103205531
    zrng(25) = 762430696
    zrng(26) = 1922803170
    zrng(27) = 1385516923
    zrng(28) = 76271663
    zrng(29) = 413682397
    zrng(30) = 726466604
    zrng(31) = 336157058
    zrng(32) = 1432650381
    zrng(33) = 1120463904
    zrng(34) = 595778810
    zrng(35) = 877722890
    zrng(36) = 1046574445
    zrng(37) = 68911991
    zrng(38) = 2088367019
    zrng(39) = 748545416
    zrng(40) = 622401386
    zrng(41) = 2122378830
    zrng(42) = 640690903
    zrng(43) = 1774806513
    zrng(44) = 2132545692
    zrng(45) = 2079249579
    zrng(46) = 78130110
    zrng(47) = 852776735
    zrng(48) = 1187867272
    zrng(49) = 1351423507
    zrng(50) = 1645973084
    zrng(51) = 1997049139
    zrng(52) = 922510944
    zrng(53) = 2045512870
    zrng(54) = 898585771
    zrng(55) = 243649545
    zrng(56) = 1004818771
```

```
RNG - 2

    zrng(57) = 773686062
    zrng(58) = 403188473
    zrng(59) = 372279877
    zrng(60) = 1901633463
    zrng(61) = 498067494
    zrng(62) = 2087759558
    zrng(63) = 493157915
    zrng(64) = 597104727
    zrng(65) = 1530940798
    zrng(66) = 1814496276
    zrng(67) = 536444882
    zrng(68) = 1663153658
    zrng(69) = 855503735
    zrng(70) = 67784357
    zrng(71) = 1432404475
    zrng(72) = 619691088
    zrng(73) = 119025595
    zrng(74) = 880802310
    zrng(75) = 176192644
    zrng(76) = 1116780070
    zrng(77) = 277854671
    zrng(78) = 1366580350
    zrng(79) = 1142483975
    zrng(80) = 2026948561
    zrng(81) = 1053920743
    zrng(82) = 786262391
    zrng(83) = 1792203830
    zrng(84) = 1494667770
    zrng(85) = 1923011392
    zrng(86) = 1433700034
    zrng(87) = 1244184613
    zrng(88) = 1147297105
    zrng(89) = 539712780
    zrng(90) = 1545929719
    zrng(91) = 190641742
    zrng(92) = 1645390429
    zrng(93) = 264907697
    zrng(94) = 620389253
    zrng(95) = 1502074852
    zrng(96) = 927711160
    zrng(97) = 364849192
    zrng(98) = 2049576050
    zrng(99) = 638580085
    zrng(100) = 547070247

End Sub

Public Function lcgrand(Stream As Integer) As Double

    Dim zi As Long
    Dim lowprd As Long
    Dim hi31 As Long

    zi = zrng(Stream)

    lowprd = (zi And 65535) * MULT1
    hi31 = (zi \ 65536) * MULT1 + lowprd \ 65536
    zi = ((lowprd And 65535) - MODLUS) + ((hi31 And 32767) * 65536) + (hi31 \ 32768)

    If zi < 0 Then zi = zi + MODLUS

    lowprd = (zi And 65535) * MULT2
    hi31 = (zi \ 65536) * MULT2 + (lowprd \ 65536)

    zi = ((lowprd And 65535) - MODLUS) + ((hi31 And 32767) * 65536) + (hi31 \ 32768)

    If zi < 0 Then zi = zi + MODLUS

    zrng(Stream) = zi

    lcgrand = (zi \ 128 Or 1) / 16777216#

End Function
```

RNG - 3

```
Sub lcgrandst(zset As Long, Stream As Integer)

    zrng(Stream) = zset

End Sub

Function lcgrandgt(Stream As Integer) As Long

    lcgrandgt = zrng(Stream)

End Function

Public Function Expon(Mean As Double, seed As Integer) As Double

' function to generate exponential variates with mean Mean via inverse cdf

    Expon = -VBA.Log(1 - lcgrand(seed)) * Mean

End Function

Public Function Uniform(Lower As Double, Upper As Double, seed As Integer) As Double

' function to generate U(Lower, Upper) variates via inverse cdf

    Uniform = Lower + (Upper - Lower) * lcgrand(seed)

End Function

Public Function random_integer(ByRef prob_distrib() As Double, seed As Integer)
    Dim i As Integer
    Dim U As Double

    'Generate a U(0,1) random variate.
    U = lcgrand(seed)

    'Return a random integer in accordance with the (accumulative) distribution
    '  function prob_distrib.
    random_integer = 1
    While U >= prob_distrib(random_integer)
        random_integer = random_integer + 1
    Wend
End Function

Public Function erlang(m As Integer, Mean As Double, Stream As Integer) As Double  '/* Erlang variate
generation
                                                                                    'function. */
    Dim i As Integer
    Dim mean_exponential As Double
    Dim Sum As Double

    mean_exponential = Mean / m
    Sum = 0#
    For i = 1 To m
        Sum = Sum + Expon(mean_exponential, Stream)
    Next i
    erlang = Sum
End Function

Public Function ErlangOneStep(n As Integer, Beta As Double, Stream As Integer) As Double
' Erlang generation in one step

    Dim U As Double
    Dim i As Integer
    Dim mean_exponential As Double
    Dim Prod As Double

    Prod = 1#
    For i = 1 To n
        U = 0#
        U = lcgrand(Stream)
        'Prod = Prod * U
        Prod = Prod * (1 - U)
```

```
    Next i

    'ErlangOneStep = -(Beta / N) * VBA.Log(Prod)
    ErlangOneStep = -Beta * VBA.Log(Prod)

End Function

Public Function GammaSpecOne(n As Integer, Stream As Integer) As Double
' generate gamma(N,1)

    Dim U As Double
    Dim i As Integer
    Dim Sum As Double

    Sum = 0#
    For i = 1 To n
        U = 0#
        U = lcgrand(Stream)
        Sum = Sum - VBA.Log(U)
    Next i

    GammaSpecOne = Sum


End Function
```