ASSIGNMENT 1:

(1) Develop a Linear Congruent Random Number Generator (L'Ecuyer, P. 1988, Communications of the ACM, vol. 31, pp. 742–774.) with two modulus algorithm. Use the generator to generate:
    a. 100 Random numbers between 1 and 1000 and plot a histogram.
    b. 10,000 Random numbers between 0 and 1 and plot a histogram
    c. What pattern do you observe.
(2) Playing Roulette. Simulate a Roulette game with a bet on numbers (ignore colors and zeros). Each bet is for $1 based on the last two numbers of your student ID. If the student ID number is 0, bet on 1. If the last two numbers of your student ID number is greater than 36, bet on 36.
    - If your number is drawn you win $36 [Profit = $35]
    - If another number is drawn you get 0. [Loss = -$1]
    a. Assume that the roulette is fair and has 36 slots. Each slot has an equal probability. Use your random number generator above and generate draws between 1 and 36. Confirm the distribution of the random numbers using a histogram.
    b. Using simulations, calculate the expected Profit/Loss for 10 plays of Roulette. Also simulate for 100 plays and 1,000 plays. Calculate the mean and standard deviation of the Profit/Loss. In each
    c. Redo the simulations for an "unfair" roulette. The payoffs are the same, but now the roulette has 37 slots (one slot is 0 and nobody can bet on that slot).
(3) Use the random number generator to simulate draws from a normal distribution. Use the inverse-normal technique show in Glasserman. Simulate 10,000 draws and draw a histogram of the distribution.
(4) Pick a stock that starts with the same letter as your first name (if you don't find a stock, use the last letter of your first name). Preferably the stock should pay dividend. Use finance.yahoo.com to find option prices on the stock of your choice. Pick an option that has a maturity between 6 and 9 months. The stock will pay at least one dividend over the life of the option. Let the strike price be equal to the stock price for the chosen maturity.
    a. Calculate the Black-Scholes value of the at-the-money call and put. Does the option price match the traded price? If there is a discrepancy, why is there a difference.
    b. Calculate the value of the at-the-money call and put option using Monte Carlo simulations. Does the simulated value match the traded price? Match Black-Scholes?

$a_0 =$	2.50662823884	$b_0 =$	-8.47351093090
$a_1 =$	-18.61500062529	$b_1 =$	23.08336743743
$a_2 =$	41.39119773534	$b_2 =$	-21.06224101826
$a_3 =$	-25.44106049637	$b_3 =$	3.13082909833

$c_0 = 0.3374754822726147$	$c_5 = 0.0003951896511919$
$c_1 = 0.9761690190917186$	$c_6 = 0.0000321767881768$
$c_2 = 0.1607979714918209$	$c_7 = 0.0000002888167364$
$c_3 = 0.0276438810333863$	$c_8 = 0.0000003960315187$
$c_4 = 0.0038405729373609$	

Fig. 2.12. Constants for approximations to inverse normal.

Input: u between 0 and 1
Output: x, approximation to $\Phi^{-1}(u)$.
$y \leftarrow u - 0.5$
if $|y| < 0.42$
  $r \leftarrow y * y$
  $x \leftarrow y * (((a_3 * r + a_2) * r + a_1) * r + a_0)/$
      $((((b_3 * r + b_2) * r + b_1) * r + b_0) * r + 1)$
else
  $r \leftarrow u$;
  if $(y > 0)$ $r \leftarrow 1 - u$
  $r \leftarrow \log(-\log(r))$
  $x \leftarrow c_0 + r * (c_1 + r * (c_2 + r * (c_3 + r * (c_4 +$
      $r * (c_5 + r * (c_6 + r * (c_7 + r * c_8)))))))$
  if $(y < 0)$ $x \leftarrow -x$
return x

Fig. 2.13. Beasley-Springer-Moro algorithm for approximating the inverse normal.

```
FUNCTION ran2(idum)
INTEGER idum,IM1,IM2,IMM1,IA1,IA2,IQ1,IQ2,IR1,IR2,NTAB,NDIV
REAL ran2,AM,EPS,RNMX
PARAMETER (IM1=2147483563,IM2=2147483399,AM=1./IM1,IMM1=IM1-1,
     IA1=40014,IA2=40692,IQ1=53668,IQ2=52774,IR1=12211,
     IR2=3791,NTAB=32,NDIV=1+IMM1/NTAB,EPS=1.2e-7,RNMX=1.-EPS)
```

Long period ($> 2 \times 10^{18}$) random number generator of L'Ecuyer with Bays-Durham shuffle and added safeguards. Returns a uniform random deviate between 0.0 and 1.0 (exclusive of the endpoint values). Call with idum a negative integer to initialize; thereafter, do not alter idum between successive deviates in a sequence. RNMX should approximate the largest floating value that is less than 1.

```
INTEGER idum2,j,k,iv(NTAB),iy
SAVE iv,iy,idum2
DATA idum2/123456789/, iv/NTAB*0/, iy/0/
if (idum.le.0) then                    Initialize.
    idum=max(-idum,1)                  Be sure to prevent idum = 0.
    idum2=idum
    do 11 j=NTAB+8,1,-1                Load the shuffle table (after 8 warm-ups).
        k=idum/IQ1
        idum=IA1*(idum-k*IQ1)-k*IR1
        if (idum.lt.0) idum=idum+IM1
        if (j.le.NTAB) iv(j)=idum
    enddo 11
    iy=iv(1)
endif
k=idum/IQ1                             Start here when not initializing.
idum=IA1*(idum-k*IQ1)-k*IR1            Compute idum=mod(IA1*idum,IM1) without over-
if (idum.lt.0) idum=idum+IM1               flows by Schrage's method.
k=idum2/IQ2
idum2=IA2*(idum2-k*IQ2)-k*IR2          Compute idum2=mod(IA2*idum2,IM2) likewise.
if (idum2.lt.0) idum2=idum2+IM2
j=1+iy/NDIV                            Will be in the range 1:NTAB.
iy=iv(j)-idum2                         Here idum is shuffled, idum and idum2 are com-
iv(j)=idum                                 bined to generate output.
if(iy.lt.1)iy=iy+IMM1
ran2=min(AM*iy,RNMX)                   Because users don't expect endpoint values.
return
END
```