ECE 3020 – Mathematical Foundations of CmpE **Spring 2017**

Programming Assignment #1
DUE March 14, 2017

You are required to complete this assignment INDIVIDUALLY.

Complete this programming assignment using a high-level procedural programming language (e.g., Java, C, C++; not MATLAB) of your choice. Carefully read the instructions below regarding printouts and other documentation; note that you do NOT submit an executable copy of your program.

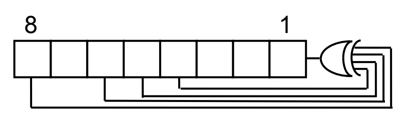
The deliverables for this programming assignment must be submitted electronically under the "Assignments" tab on T-Square no later than 11:00 pm on the due date listed above. A single pdf file is the preferred format. Clearly PRINT your name at the top of each sheet. You are responsible for ensuring that your submitted file(s) are readable.

While hardware modules exist that use physical phenomena to generate truly random numbers, most computing systems use either hardware or software pseudo-random number generators (PRNGs) that produce sequences of values that appear to be random. PRNGs can be initialized to a specific value, allowing the same sequence of (seemingly) random values to be generated repeatedly, which is helpful in program debugging. Most programming languages and operating systems provide a built-in function to generate random numbers uniformly distributed in the range (0, 1); these values then can be either (a) appropriately scaled to yield a desired continuous random function or (b) divided into proportionally-sized ranges for discrete random functions.

With PRNGs, the obvious question of "how random are they, really?" can be partially answered by applying various mathematical measures to the generated sequences and comparing the results with those of truly random sequences. Obvious measures include the mean value and, for discrete functions, how often each value occurs. Such tests are of limited value, however, since a counter that repeatedly cycles through the numbers 1 through 6 would generate values with the same frequency as a true six-sided die, even though it clearly is not random.

Other mathematical measures are also used to measure randomness; for example, when flipping a coin, the actual frequency of various runs of consecutives heads or tails can be compared to the expected frequency of occurrence. This assignment compares PRNGs using the simple mathematical measures of **mean** and **standard deviation**, considering both short and long sequences of random values.

Linear Feedback Shift Registers (LFSRs) are commonly implemented in hardware for applications including low-cost counters (with an unusual sequence), random number generators, coding, and signature generation (e.g., checksums) for blocks of data. The circuit to the right illustrates an 8-bit LFSR; the characteristic polynomial defines the locations of the "taps" (bits that are XORed to compute the next least-significant-bit). For appropriate polynomials, the LFSR produces a maximum-length cycle of $2^n - 1$ values, excluding only the all-zero value. Other hardware implementations are also used in various applications, but this configuration is easily simulated in software. Dividing the LFSR value by 2^n yields a result in the range (0, 1).



Characteristic polynomial:
$x^8 + x^6 + x^5 + x^4 + 1$

Note that LFSR bits are normally numbered 1 through n, rather than 0 through n-1 as for binary numbers.

The n-input XOR performs a bit-wise XOR of its inputs, so the XOR output equals 1 when an odd number of its input bits are equal to 1.

Numerous categories of software algorithms also exist for generating random sequences. A simple one is the Linear Congruential Generator (LCG), which is defined by the relation: $X_{n+1} = (\mathbf{a}\, X_n + \mathbf{c}) \bmod \mathbf{m}$. The resulting value or a specified subset of bits is used to generate the output value. (In many cases lower-order bits are not used in forming the output value, since they may not exhibit the same quality of mathematical randomness.)

1) Using the high-level programming language of your choice, write a program that compares your system's built-in random number generation function with LFSR and LCG implementations of pseudo-random number generators. Program specifications:

   a) Write functions RAN_LFSR and RAN_LCG that generate random 24-bit values using the methods described above. Each time one of these functions is called, it should use the appropriate algorithm to generate the next random number in its sequence. Use the following parameters for the LFSR and LCG:

      LFSR: $x^{24} + x^{23} + x^{22} + x^{17} + 1$

      LCG: a = 1140671485 (0x43FD43FD), c = 128201163 (0xC39EC3), $m = 2^{24}$; use all 24 bits to generate the value

      After generating the next 24-bit value, the function should return a real number in the range (0, 1) that is generated by dividing its 24-bit value by 2^{24}.

   b) Initialize the LFSR and the LCG to 0xFFFFFF (24 bits, all '1's) when the program starts.

   c) Generate 10,000 sets of random values using the built-in random number function, RAN_LFSR, and RAN_LCG. As the program runs, it should generate the following results:

      i. A table, similar to the following, showing the **first 25** random values generated by each method. *(The following are not the actual values that should be generated.)*

Built-In Function	**LFSR**	**LCG**
0.038965	0.703815	0.416631
0.405642	0.007629	0.033824
0.543324	0.915259	0.210261

      ii. A summary table, similar to the following, listing the **mean** and **standard deviation** of the random values generated by each method. Print this table after 25, 50, 100, 500, 1000, 5000, and 10000 sets of random numbers. The first row in the summary table should correspond to the random numbers from part (i) above, allowing you to verify your calculations. *(The following are not the actual expected values.)*

# of values	**Built-In Function**		**LFSR**		**LCG**	
generated	**mean**	**std dev**	**mean**	**std dev**	**mean**	**std dev**
25	0.543	0.163	0.502	0.278	0.458	0.106
50	0.493	0.069	0.521	0.135	0.503	0.092
100						
500						
1000			etc.			
5000						
10000						

2) Briefly summarize and interpret the results obtained, comparing your results to the theoretical values. (The theoretical **mean** is obviously 0.500; can you figure out the theoretical value of the **standard deviation**?) Are these three options for generating random numbers essentially equivalent, or do they have distinctive characteristics? Consider both what happens initially and the long-term trends. If possible, explain any observed differences between the LFSR and LCG implementations.

Submit the following materials: (A) A listing of your program source code (.pdf or .txt). **(B)** The program output (captured from your computer screen or printed to a file), showing the execution of your program and its generation of both the table of initial random values and the summary tables of statistics, as described above. **(C)** Your summary and interpretation of the results.