Programming Assignment #3
DUE March 27, 2019

You are required to complete this assignment INDIVIDUALLY.

Complete this programming assignment using a high-level procedural programming language (e.g., Java, C, C++) of your choice, MATLAB, or an Excel spreadsheet (similar to the examples shown in lecture). Carefully read the instructions below regarding printouts and other documentation; note that you do NOT submit an executable copy of your program **unless you implement your simulation using Excel**.

The deliverables for this assignment must be submitted electronically on the Canvas "Assignments" page 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 and successfully uploaded to Canvas.

Probabilistic algorithms and/or stochastic simulation are often used to estimate values or predict the likelihood of specific events, particularly if direct calculation is algorithmically complex or involves interactions that are difficult to formally characterize, but easy to simulate computationally.

Consider the following "game" that uses a single fair die; i.e., the values 1 through 6 are equally likely for each roll of the die: *Starting with a total of zero (0), roll the die and add its value to the total, repeating until the total of all the rolls first exceeds twelve (12). For example, the sequence of five rolls 6 – 2 – 1 – 2 – 5 results in a final total of 16.*

(A) Before writing the program, answer the following questions, which may help you develop the simulation. These questions can be answered by observation or simple reasoning, with no significant calculations or computation needed.

i. What are the possible final totals? Why?

ii. Which final total is most likely? Why?

iii. What are the minimum and maximum number of rolls of the die required to achieve a valid final total? Why?

(B) Develop a computational solution (program or spreadsheet) that simulates this game using a fair die and estimates the following: *for each possible final value, (i) what is the probability of achieving this value and (ii) what is the average number of rolls required to achieve this value*.

(C) Your simulation should generate 25,000 trials and produce the following outputs: (i) Results of the first ten (10) trials listing the value of each die rolled, the final total, and the number of rolls. (This will allow verification of your simulation.) (ii) A table summarizing the results (as defined in part B above) after 25, 50, 100, 250, 500, 1000, 2500, 5000, 10000, 15000, 20000, and 25000 trials.

(D) Repeat the simulations from steps B and C, but with a non-uniform die that generates each of the possible outcomes with the following specified probabilities:

Result:	1	2	3	4	5	6
Probability:	0.05	0.15	0.30	0.30	0.15	0.05

(E) Briefly summarize your observations from these simulations, including issues such as the number of trials required for different values to converge, the stability of the results for various calculations, and any other observations regarding the simulation process or results.

Submit the following materials, preferably combined into a single .pdf file:

(a) Answers and brief explanations for the questions.

(b) Your program source code listing (.pdf or .txt) or Excel file (.xls).

(c) Program printout (or one-page portion of Excel spreadsheet) with the specified simulation results.

(d) Brief (1 or 2 paragraphs) written summary discussing the indicated issues.

Hints for generating random values to meet specific application needs:

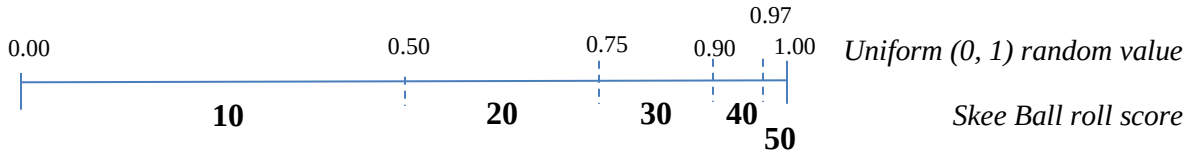
Most programming languages and computing systems include the ability to generate random values, typically in the form of a uniformly distributed real number between 0 and 1. By appropriately scaling and shifting this value, a uniform distribution between any two real numbers can be easily obtained. Non-uniform distributions, such as a Gaussian (normal) function, may be derived by treating the system-generated value as the cumulative probability density and working backward to calculate the desired function parameter.

Many applications require discrete random values distributed according to specified probabilities. For equally likely values, the uniform random value can be scaled and then converted to an integer. To avoid issues related to whether or not the 0 and 1 boundary values may occur, the scaled value can be shifted by 0.5 and rounded to generate a series of equally likely integers. For example, the roll of a single die can be simulated by the following calculation:

D = ROUND (6 * RAN() + 0.5)

which takes a random value in the range (0, 1), maps it to a value in the range (0.5, 6.5), and then rounds it to an integer in the range [1, 6].

If an application requires discrete random values with unequal probabilities, values can be mapped from a uniform space to a non-uniform space using CASE or IF-THEN-ELSE instructions or the Excel HLOOKUP function. For example, uniform random values over (0, 1) can be mapped to the Skee Ball distribution from Homework #5 as illustrated below:



Spreadsheet programs like Excel often include built-in functions to generate random numbers according to various sets of parameters, such as those discussed in the above paragraphs.