MAT 340 - CODING ASSIGNMENT #4
due Friday, April 19, 2019 at 8:00PM.

OBJECTIVE: Students will model various random processes of interest.

GRADING: The assignment entirely extra credit, worth an extra 4% of your course grade.

INSTRUCTIONS:

- Students must work **individually** and submit their own code, but they may ask questions and clarification from classmates and the instructor.
- Students must submit their projects on Moodle.

SUBMIT THE FOLLOWING:

- An executable. This should be able to run on a clean machine, please compile it accordingly (see the MAT 340 forum for a build tip in Visual Studio).
- A read-me file explaining how to run your code. Include your name, course, section and semester in the header of your file.

PROJECT:

1. **The Hat Problem:** N people place their hats in a pile. Each person is then given a hat, chosen at random from the pile. How may people get their own hat back? Write a program that does the following:
   - Input the number of people N.
   - Input the number of trials m, with $m \geq 100$.
   - In a loop of m trials:
     - Produce a random permutation of $(1, 2, \cdots, N)$, let's call it $(p_1, p_2, \cdots, p_N)$
     - Count how many people got their hat back, that is, for how many k is $k = p_k$?
   - Estimate the expected number of hats returned to their owners.
   - Estimate the variance for the number of hats returned to their owners (Hint: compute the second moment first.)

2. **The Stepping Stone model:** Start with an $n \times n$ array of squares, where each square is initially any one of k different colors ($k < n^2$). For each step, a square is chosen at random. This square then chooses one of its eight neighbors at random and assumes the color of that neighbor. To avoid boundary problems, imagine making the array into a cylinder by gluing the top and bottom edge together, and then making the cylinder into a doughnut by gluing the two circular boundaries together. Then each square in the array is adjacent to exactly eight other squares. Run a simulation and see how the colors change. One should be able to adjust the size of the grid, the number of colors, and for how long to run the simulation.

3. **Longest increasing (consecutive) subsequence**: A random permutation of numbers 1 to N is a random arrangement of numbers $1, 2, 3, \ldots, N$. The goal of this assignment is to find the expected length of the longest increasing (consecutive) subsequence. For example, suppose the sequence has length 8 and a random permutation is 14562378. Then, the increasing (consecutive) subsequences of this random permutation are $\{1, 456, 23, 78\}$, with the longest being 456, of length 3.

   Write a program that does the following:

   - Input the length of the sequence N.
   - Devise a way to order the $N!$ arrangements of numbers $1, 2, 3, \ldots, N$.
   - In a loop of 100 trials:
     - Pick one of the $N!$ arrangements, uniformly at random.
     - Find the length of the longest increasing (consecutive) subsequence.
   - Output the average length of the longest increasing (consecutive) subsequence of a random permutation of length N.

4. **Random walk (intersection)**: A simple d-dimensional random walk is a Markov chain that at each step moves one unit in one of the $(2d)$ directions with equal probability. Estimate the probability that two random walk paths intersect.

   Write a program that does the following:

   - Input the dimension d.
   - Input the number of trials M, with $M \geq 1,000,000$
   - Input the length of each random walk n, with $n \geq 400,000$
   - In a loop of M trials:
     - Start a random walk RW1 at $\vec{1} = (1, \cdots, 1)$, let it run for n steps and record its path.
     - Start another random walk RW2 at $-\vec{1} = (-1, \cdots, -1)$ and run it until it hits the path of RW1 or for n steps, whichever comes first.
     - Record if RW2 hit the RW1 path.
   - Let $M(n)$ be the number of pairs of random walks that have an intersection.
   - Compute and output the "intersection exponent" $\xi = \dfrac{\log M - \log(M - M(n))}{\log n}$.

   This simulation will run slow. Test your code with smaller values for M and n first.