**CSCI 236 Python Programming Fall 2018**

**Final Exam**

1. Word Count. A common utility on UniX/Linux systems is a small program called ''we." This program analyzes a file to determine the number of lines, words, and characters contained therein. Write your own version of wc. The program should accept a file name as input and then print three numbers showing the count of lines, words, and characters in the file in one line using string formatting.
2. Write a function that calculates the distance in miles between to cities, given the latitude and longitude coordinates of each.

Then, write a program that prompts for the coordinates of two cities, uses the function to calculate the distance, and outputs the result.

Use variable-length arguments for setting different rounding and accuracy (one decimal place, two decimal places, round to the closest integer).

1. Assume that we have a 5 by 3 matrix A = [[1,2,3],[4,5,6],[7,8,9],[10,11,12],[13,14,15]]. Output the product of two matrices A\*A’, where A’ is the transpose of A. Implement it using two different approaches. Hint: you might consider using nested list comprehension. You can find examples on matrix multiplication here: https://www.mathsisfun.com/algebra/matrix-multiplying.html
2. The “Monty Hall Problem” (https://en.wikipedia.org/wiki/Monty\_Hall\_problem) is a classic introduction to game theory and probability simulations that comes from a live TV game show in the 1960s. A contestant is faced with three doors. Behind one door is a very large prize (say, a car). The contestant tries to guess which door has the prize. We already know that mathematically the contestant should win 33% of the time. Write a program that uses a Monte Carlo simulation to prove it. For each simulation run, the program should use a variable to represent the randomly chosen winning door number. The program should then randomly select a door number from one to three to represent the contestant’s choice. Finally, display the winning percentage after a large number of simulation runs, and confirm that it is very close to 33%.

The second part of the “Monty Hall Problem” is the most interesting. After selecting a door, but before the prize door is revealed, the contestant is shown a losing door. The contestant now has the option of switching to the other, as-yet unopened door. Should the contestant stick with their original choice, switch to the other unopened door, or does it not matter? Modify your program from the first part to always choose to switch to the other door, and display the winning percentage after a large number of simulation runs. What would you conclude to be the best strategy? For a complete description of this problem, see <https://en.wikipedia.org/wiki/Monty_Hall_problem>.

1. Write a graphical bingo game. A typical bingo game is a 5-by-5 grid of numbers that can have randomized values from 1 to 75. The computer randomly selects numbers from 1 to 75, and the player clicks on a matching square to “cover” the square. The player wins when they have covered squares in a horizontal row, vertical column, or one of the two diagonals. Your program should ensure that the user can only click on a square containing a number that was just picked, and it should automatically identify a winning card. Use exception handling statements (raise an exception) if possible. Let the user play five bingo cards at a time.