15110 Summer 2018
Problem Set 9

Name: ____

Andrew ID: ____

Instructions

- Type or neatly write the answers to the following problems.
- Save or scan this file as a pdf and submit to Gradescope

Exercises

1. (4 points) This question deals with generating random numbers using a linear congruential generator (LCG). Here's the formula for an LCG:

$$x_{i+1} = (a \times x_i + c) \mod m$$

Recall the conditions that must hold for an LCG with parameters a, c, and m to have its maximum period of m:

i. c and m must be relatively prime;
ii. a-1 must be divisible by every prime factor of m; and
iii. if m is divisible by 4, then a-1 must be divisible by 4.

For instance, $a = 5$, $c = 7$, and $m = 16$ meet these conditions (remember, 2 is the only prime factor of a power of 2):

i. the factors of 7 are 1 and 7, the factors of 16 are 1 and 2, so 7 and 16 are relatively prime;
ii. 4 is divisible by 2; and
iii. both 16 and 4 are divisible by 4.

For this problem, think about a linear congruential generator where m is a power of 2 (like 8, 16, 32, etc.).

A. (1 point) When this kind of LCG has a period equal to *m*, is *c* odd or even? Why?

B. (1 point) When this kind of LCG has a period equal to *m*, is *a* odd or even? Why?

C.

(2 points) No matter how large *m* is, if it is a power of 2, there is something we can predict about the next output value of the LCG, based on the current output. That is, if x_i and x_{i+1} are two numbers produced by the LCG in sequence, and we know x_i, we can predict something about x_{i+1}. This prediction is based on the laws of simple arithmetic having to do with even and odd numbers What is it? Why does this happen?

Bonus: Consult Wikipedia or some other source and give one technique for using the output of this kind of LCG that mitigates this problem

2. (1 pt) Assume that the random number generator `randint(m,n)` is *uniformly distributed*. This means that the probability of generating any number between m and n is the same.

   Suppose that Alice simulates the roll of a pair of dice by defining the roll function below, calling it twice and adding the results:

   ```
   def roll():
       return randint(1,6)
   ```

   Bob realizes that the roll of a pair of dice results in a sum of 2 through 12, inclusive, so he simulates the roll of a pair of dice by defining the roll function below, calling it only once:

   ```
   def roll():
       return randint(2,12)
   ```

   Are these equivalent in terms of their behavior over time as we generate roll after roll? Why or why not?

3. (2 points)

   Consider the following code written for a simple game of Dungeons & Dragons. In this game, the player rolls one 20-sided die 20 times. Each time, if the value thrown is 20, we have a critical hit and keep track of it by adding 1 to critHitTotal. Similarly, if the value thrown is 1, we have a critical miss and keep track of it by adding 1 to critMissTotal

   There is a logical error in the program for this game below. Explain what is wrong and show how to correct it.

```
from random import randint

def roll():
    return randint(1,20)

def game():
    critHitTotal = 0
    critMissTotal = 0
    for i in range(20):
        if roll() == 20:
            critHitTotal += 1
        elif roll() == 1:
            critMissTotal += 1
    return (critHitTotal, critMissTotal)
```

4. (3 points) Think about the following program:

```
from random import random
from math import sqrt

def mystery(n):
    hits = 0
    for i in range(n):
        x = random()
        y = random()
        r = sqrt(x**2 + y**2)
        if r <= 1.0:
            hits = hits + 1
    area = hits/n
    return 4*area
```

This is an example of a Monte Carlo method, where each "experiment" picks a random point with *x* and *y* coordinates
between 0 and 1. What geometric figure does `area` correspond to? What quantity does this function approximate?