Foundations of Applied Math
HW #10
Due Friday, Nov. 13 by 7 am

Reminder You need to turn in a .zipped folder that contains your .tex file, your image files, your python files and the tex file must compile. Rename the .tex file: HW9_YourLastName.tex and call the folder which you will compress: HW9_YourLastName

1. Given the code shown below that was posted in the second random walk video on Nov. 10 alter the code so that it still prints to the screen but that it also produces a bar plot like I showed you on Nov. 12.

```
import random
import matplotlib.pyplot as plt
import numpy as np

def animate(x,y):
    plt.plot(x,y,'r.')
    return()

def randomwalk(n):
    x=0
    y=0
    #animate(x,y)
    for i in range(n):
        step=random.choice(['N','S','E','W'])
        if step == 'N':
            y+=1
        elif step=='S':
            y+=-1
        elif step=='E':
            x+=1
        else:
            x+=-1
        #animate(x,y)

    return(x,y)


N=10000
transitPercent = np.zeros(41)
for walklength in range(41):
    notransit=0
    for trials in range(N):
        pt=randomwalk(walklength)
        dist=abs(pt[0])+abs(pt[1])
        if dist<=4:
            notransit=notransit+1
    transitPercent[walklength] = notransit/N
    print("The prob. that I can walk home is ", notransit/N, " for a walk of size "
plt.xlabel("Number of Walks")
plt.ylabel("Percent")
plt.bar(range(41), transitPercent)
plt.show()
```

2. Why do the even numbered walk lengths have higher probabilities than the odd numbered walk lengths in the problem above?
   Running the code above with the maximum distance set to an even number, we find that the even random walks seem to have a higher probability than the odd random walks. With the maximum distance set to an odd number, we find that the odd random walks have a higher chance than the even random walks.

3. Given the region trapped between $y = x^2$ and $y = 6 - x$ and the x and y axes, write python code to find the area of this trapped region using a Monte Carlo simulation.
   number3.py is included.

4. Using a Monte Carlo simulation, write python code to approximate the volume trapped between $z = 9 - x^2 - y^2$ and $z = 2x^2 + 2y^2$ using a Monte Carlo simulation.
   number4.py is included.

5. Suppose you are playing a game in which you are getting tickets that have winning letters: 65% of the tickets contain a the letter "A", 10% of the tickets contain the letter "B", and 25% contain a "C" A participant in the game wins a prize by obtaining all three letters A, B, and C. So you have to buy at least 3 tickets to possibly win. Write a python program that uses Monte Carlo simulation that could be used to determine how many tickets you expect to buy to win a prize. number5.py is included.