CS1110 Lab 5: Practice for A4 (Mar 8-9, 2016)

First Name: _____ **Last Name**: _____ **NetID:** _____

The lab assignments are very important. Remember this: *The lab problems feed into the assignments and the assignments define what the exams are all about.*

Start *before* your lab meets.

We recommend spending an hour or two on the lab *before* coming to your section, so you can use your in-person time to ask questions most efficiently.

Also, this strategy of starting beforehand increases your chances of checking in at your lab section, which will probably take less time than waiting in line at consulting hours!

Getting credit

Complete all required blank boxes and lines on this handout. When you are finished, show your written answers to one of the CS 1110 lab staff in your section on March 8-9 or in any consulting hours up to and including March 14 (earlier days have shorter lines). The staff member will ask you a few questions to make sure you understand the material, and then swipe your Cornell ID card or directly make a notation in CMS to record your success. This physical piece of paper is yours to keep.

Getting set up

From the Lab webpage, download and unzip `Lab5.zip` into a folder named (for example) `Lab_5`. In the command shell, navigate the file system so that this folder is THE CURRENT WORKING DIRECTORY.

Solutions

1 Random Simulation Basics

The following script produces 4 lines of output:

```
# ShowRand.py
from random import randint as randi
from random import uniform as randu

N = 100000
count=0
for k in range(N):
    x = randu(0,100)
    if 10<=x<=30 or 40<=x<=70:
        count +=1
print float(count)/float(N)

count=0
for k in range(N):
    x = randu(0,100)
    if 10<=x<=50 and  20<=x<=60:
        count +=1
print float(count)/float(N)

count=0
for k in range(N):
    x = randi(1,20)
    if x%2==1:
        count +=1
print float(count)/float(N)

count=0
for k in range(N):
    x = randi(1,20)
    if 13<=x<15:
        count +=1
print float(count)/float(N)
```

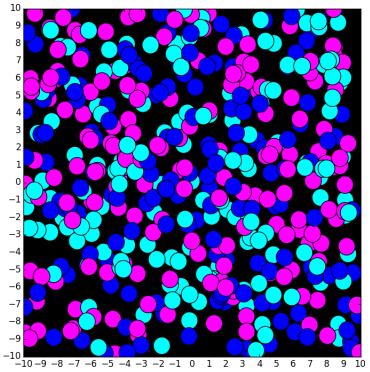
Fill in the following table:

		I Think it's This	**Python Says**	**Notes**
1	`First Output Line`			
2	`Second Output Line`			
3	`Third Output Line`			
4	`Fourth Output Line`			

`First Line: 0.5  Second Line = 0.3  Third Line = 0.5  Fourth Line = 0.10`

2 Paintball

Run the module `Paintball.py` and observe that it randomly places 500 color disks in the figure window:



Look at the computation of `x` and `y` inside the `for` loop. Show how to modify these assignment statements so that no paintball is "cut off" by the edge of the figure window?

```
x = randu(-n+r,n-r)
y = randu(-n+r,n-r)
```

As it stands, the color is Magenta with probability 1/3, Cyan with probability 1/3, and Blue with probability 1/3. How could you change

```
rc = randi(1,3)
if rc%3==0:
    c = MAGENTA
elif rc%3==1:
    c = CYAN
else:
    c = BLUE
```

so that we draw a Magenta disk with probability 1/2, a Cyan disk with probability 1/3, and a Blue disk with probability 1/6? Hint: `randi(1,6)`.

```
rc = randi(1,6)
if rc<=3:
    c = MAGENTA
elif rc == 4 or rc == 5:
    c = CYAN
else:
    c = BLUE
```

3 Random Walk

Here is code that simulates a random walk inside a square with corners (L, L), $(L, -L)$, $(-L, L)$, and $(-L, -L)$:

```
x = 0
y = 0
T = ''
while abs(x) < L and abs(y) < L:
     # The robot has not reached the edge
     r = randi(1,4)
     if r==1:
       # Hop North
       y = y + 1; T = T + 'N'
     elif r==2:
       # Hop East
       x = x + 1; T = T + 'E'
     elif r==3:
       # Hop South
       y = y-1; T = T + 'S'
     else:
       # Hop West
       x = x-1; T = T + 'W'
 # Code Here
```

The robot starts at (0,0) and hops its way to the edge of the square. A hop is either one unit North, one unit East, one unit South, or one unit West. The string `T` "records" the itinerary. For example, if `T` is `'NENSWSWE'` then we know that on the third hop the robot headed south because the value of `T[3]` is `'S'`. Assume that `L` is a positive integer and that `T` has been produced by the `while` loop. Add code after the `Code Here` comment so that the number of hops that the robot needed to reach the edge is assigned to `nSteps`.

```
nSteps = len(T)
```

Add code after the `Code Here` comment so that the xy coordinates of the robot's final position are printed out. (Hint. Think about `T.count('N')`, `T.count('E')`, `T.count('S')`, and `T.count('W')`.

```
xFinal = T.count('E') - T.count('W')
yFinal = T.count('N') - T.count('S')
```

As it stands the simulation has the robot hopping until it reaches an edge of the square. What can you say about the simulation if we change the `while`-loop condition to `abs(x) < L or abs(y) < L`?

```
Must have
              |x| >= L   and |y| >= L
```

4 Hexagons

In Assignment 4 you will need to work with the function `NeighborCenter` that is defined in the given module `HexTools.py`. This problem gives you experience with that function and gets you thinking about the hex-cells that are part of the scene. Hex-cells can be drawn with `DrawHexCell`. Here are the specifications for these functions:

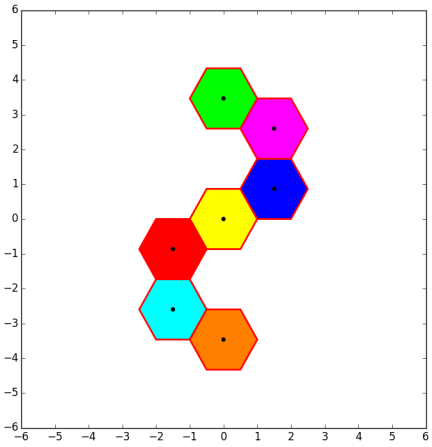
```
def DrawHexCell(a,b,FillColor=None):
    """
    Displays a regular hexagon with side one and center (a,b).
    The hexagon is colored according to FillColor.

    PreC: a and b are numbers, FillColor is an rgb list.
    """

def NeighborCenter(a,b,i):
    """ Returns floats u and v with the property that
    (u,v) is the center of the i-th hex-cell neighbor.

    PreC: a and b are floats and (a,b) is the center of a hex-cell.
    i is an int that satisfies 1<=i<=6.
```

Read §1.3 in the A4 handout for details on what we mean by "the i-th hex-cell neighbor". Below you see a seven hex-cells that are to be produced by the code on the right. However, the code on the right is incomplete. Your task is to fill in the missing pieces. Each missing piece is a *single* call to `NeighborCenter`.



```
DrawHexCell(0,0,FillColor=YELLOW)
x,y = NeighborCenter(0,0,2)
DrawHexCell(x,y,FillColor=BLUE)
x,y = NeighborCenter(x,y,1)
DrawHexCell(x,y,FillColor=MAGENTA)
x,y = NeighborCenter(x,y,6)
DrawHexCell(x,y,FillColor=GREEN)
x,y = NeighborCenter(0,0,5)
DrawHexCell(x,y,FillColor=RED)
x,y = NeighborCenter(x,y,4)
DrawHexCell(x,y,FillColor=CYAN)
x,y = NeighborCenter(x,y,3)
DrawHexCell(x,y,FillColor=ORANGE)
```

You can check things out with `SomeHexTools.py`