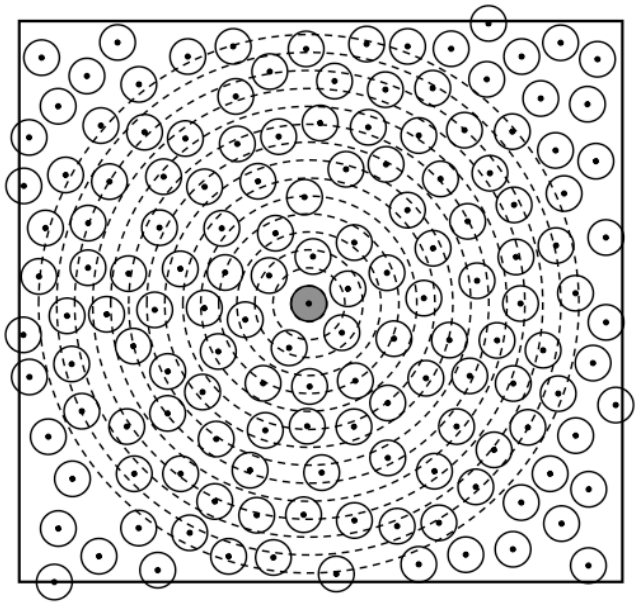
2-2. Make a xerox reproduction of Fig. 2-11 below that represents the atoms in a amorphous solid and, using a compass, manually calculate *g(r)* for a single ensemble using the dark particle as the central particle. Do this with a *dr* no larger than the particle radius, *b*. Plot your result and identify the first and second coordination spheres.

2-2. Solution:

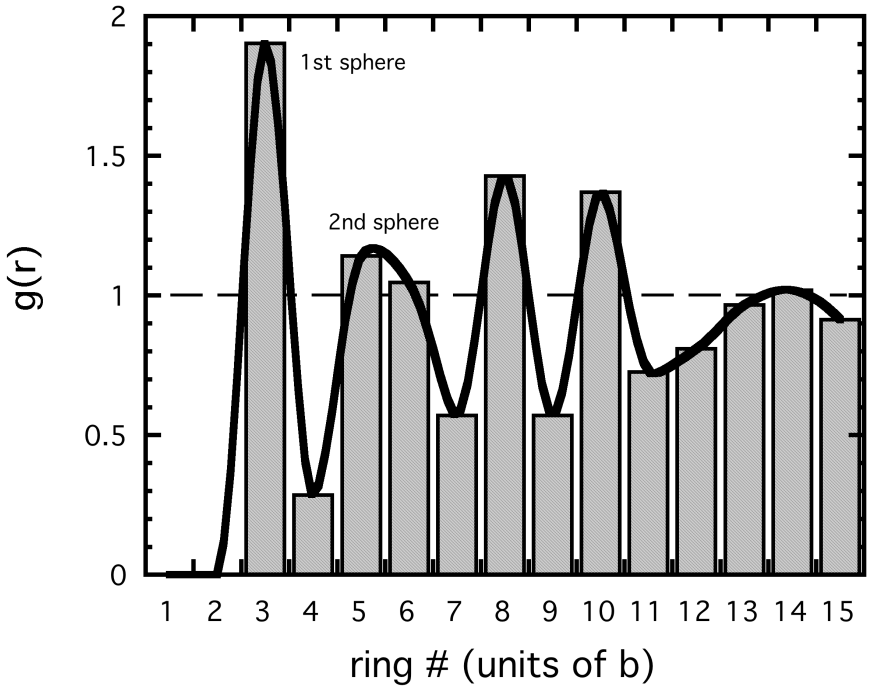
Draw rings about the central particle like so:



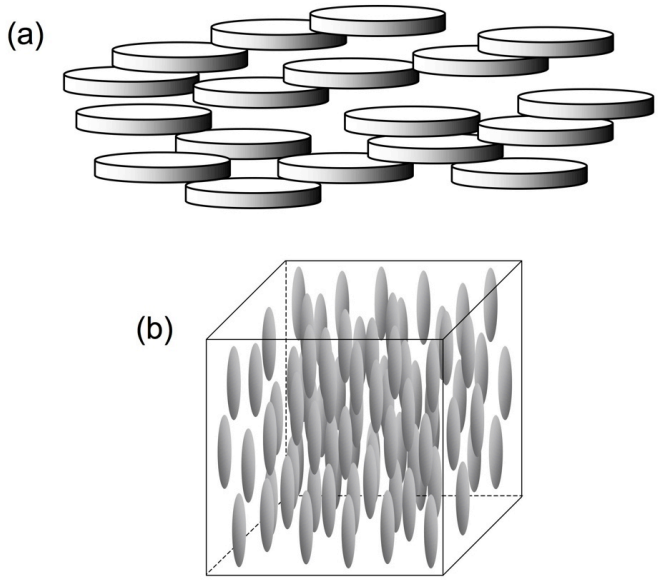
Count the number of particles centers found in each ring of size dr (here equal to b) and make a table:

	ring #	dN	g(r)
0	1.0000	0.0000	0.0000
1	2.0000	0.0000	0.0000
2	3.0000	5.0000	1.6968
3	4.0000	1.0000	0.25453
4	5.0000	5.0000	1.0181
5	6.0000	5.5000	0.93326
6	7.0000	3.5000	0.50905
7	8.0000	10.000	1.2726
8	9.0000	4.5000	0.50905
9	10.000	12.000	1.2217
10	11.000	7.0000	0.64788
11	12.000	8.5000	0.72115
12	13.000	11.000	0.86147
13	14.000	12.500	0.90902
14	15.000	12.000	0.81448

For 2D case here, $g(r) = dN/(2\pi r dr)\langle n\rangle = dN/(2\pi m b^2)\langle n\rangle$. Including the central atom, there are 98.5 centers inside the m = 15th ring of area $\pi(15b)^2$. A histogram of the g(r) (for only this one ensemble) looks like this:



2-3. Figure 2-12 shows the nematic phases of two liquid crystals: (a) a discotic liquid crystal and (b) a lipid liquid crystal. For each of these partially amorphous systems, discuss the symmetry properties including both translational and rotational symmetries.



2-3. Solution:

Note first that only an average symmetry of any kind will exist in this system. By definition, amorphous solids do not posess symmetry in the same way that a normal crystal does.

Also note that to the extent that either system has symmetry, they are identical, one is just an elongated, rounded version of the other. "Average" symmetries are as follows:

- Infinite translational Symmetry (assuming a sufficiently large volume, of course)
- Infinite rotational symmetry around the z-axis of the disks/lipids
- 180 degree rotations (two-fold rotational symmetry) about the other two axis.

6-2. Show that the reciprocal space of the BCC lattice forms an FCC lattice.

6-2. Solution:

The unit lattice vectors for the primitive BCC lattice are provided in Fig. 1-9 of Chapter 1 as:

$\vec{a}_1 = (a/2)\left(\hat{x} + \hat{y} - \hat{z}\right)$

$\vec{a}_2 = (a/2)\left(\hat{x} - \hat{y} + \hat{z}\right)$ and the transformation described by Eq. 6.9 which yields:

$\vec{a}_3 = (a/2)\left(-\hat{x} + \hat{y} + \hat{z}\right)$

$\vec{b}_1 = (2\pi/a^3)(a^2)\left(-\hat{x}/2 - \hat{y}/2\right)$

$\vec{b}_2 = (2\pi/a^3)(a^2)\left(-\hat{x}/2 - \hat{z}/2\right)$, equal to the FCC lattice (scaled by 2π/a).

$\vec{b}_3 = (2\pi/a^3)(a^2)\left(-\hat{y}/2 - \hat{z}/2\right)$

6-4. The primitive cell of the hexagonal lattice can be defined by the following lattice vectors:

$$\vec{a}_1 = \frac{\sqrt{3}a}{2}\hat{x} + \frac{a}{2}\hat{y}, \quad \vec{a}_2 = -\frac{\sqrt{3}a}{2}\hat{x} + \frac{a}{2}\hat{y}, \quad \vec{a}_3 = c\hat{z}.$$

(a) Show that the volume of the primitive cell is $\frac{\sqrt{3}}{2}a^2c$.

(b) Determine the corresponding lattice vectors describing a primitive cell of the reciprocal lattice and demonstrate that the hexagonal lattice is its own reciprocal (aside from a rotation).

6-4. Solution:

(a) the volume is given by $V = \left|\vec{a}_1 \cdot (\vec{a}_2 \times \vec{a}_3)\right| = \left|\vec{a}_1 \cdot (ac/2)(\hat{x} + \sqrt{3}\hat{y})\right| = \sqrt{3}a^2c/2$.

(b) the transform given by Eq. 6.9 produces

$$\vec{b}_1 = (4\pi/\sqrt{3}a^2c)\left[(ac/2)\hat{x} + \left(\sqrt{3}ac/2\right)\hat{y}\right] = (4\pi/\sqrt{3}a^2)\left[(a/2)\hat{x} + \left(\sqrt{3}a/2\right)\hat{y}\right]$$

$$\vec{b}_2 = (4\pi/\sqrt{3}a^2c)\left[-(ac/2)\hat{x} + \left(\sqrt{3}ac/2\right)\hat{y}\right] = (4\pi/\sqrt{3}a^2)\left[-(a/2)\hat{x} + \left(\sqrt{3}a/2\right)\hat{y}\right]$$

$$\vec{b}_3 = (4\pi/\sqrt{3}a^2c)\left[\left(\sqrt{3}a^2/2\right)\hat{z}\right] = (2\pi/c)\left[\hat{z}\right]$$

or $\vec{b}_1 = \frac{2\pi}{\sqrt{3}a}\hat{x} + \frac{2\pi}{a}\hat{y}, \quad \vec{b}_2 = -\frac{2\pi}{\sqrt{3}a}\hat{x} + \frac{2\pi}{a}\hat{y}, \quad \vec{b}_3 = \frac{2\pi}{c}\hat{z}$. These are similar to the original space lattice except that the x-components are now shorter than the y-components (as would occur if rotated about the z-axis).

Problem 5 Solution

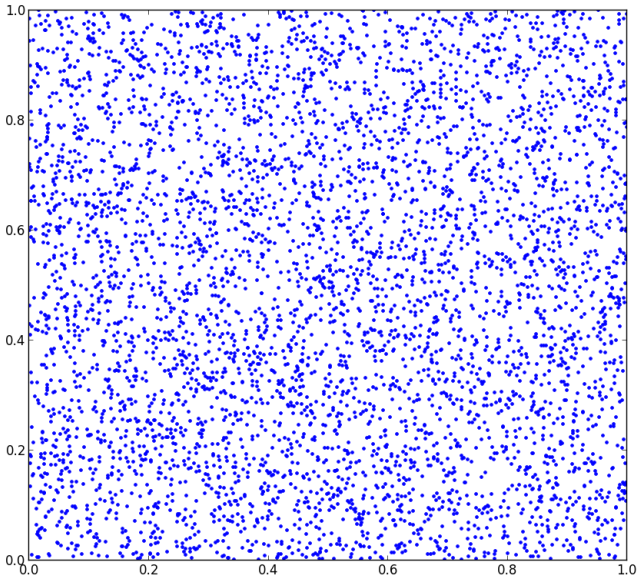
```
#!/usr/bin/python
#Matthew Lawson, written as a solution for phsyics 140a hw 2 pr. 5
#This program calculates the first five moments of a given number of
#random numbers uniformly distributed on [0,1]

import numpy as np
import scipy as sp
import matplotlib as mpl

def hw2pr5(n_moments, n_rand_numbers):
    """
    This method takes n_moments, an int, and interprets it
    as the number of moments to calculate. n_rand_numbers is interpreted as
    the number of random numbers to generate when calculating moments. It should
    also be an int.
    """

    moments = [] #initialize a list to hold the computed moments
    for i in range(1, n_moments+1):
        total = 0 #Initialize the sum for this set of moments
        for j in range(n_rand_numbers):
            total = total + np.random.random()**i
        moments.append(total/n_rand_numbers)
    #print(moments)
    return moments

def better(n_moments, n_rand_numbers):
    """
    A cleaner and faster version of the above method.
    Runs about 10x faster, takes only 2 lines.
    """

    # create an array of random numbers
    rands = np.random.random(n_rand_numbers)
    #raises each element of the array to the right power, then sums them.
    moments = [(rands**(i+1)).sum()/n_rand_numbers for i in range(n_moments)]
    #print(moments)
    return moments
```

Output:

```
In [20]: hw2_pr5.better(5, 1000000)
Out[20]:
[0.49942855820812193,
 0.33271797775945383,
 0.24940736674742567,
 0.19944817479494983,
 0.16615839658034565]
```

Problem 6 Solution

```
#!/usr/bin/python
#Matthew Lawson, written as a solution for phsyics 140a hw 2 pr. 6
#This program generates and plots pairs of
#random numbers uniformly distributed on [0,1]

import numpy as np
import scipy as sp
import matplotlib as mpl
import matplotlib.pylab

def hw2pr6(n_pairs):
    """
    generates and plots n_pairs ordered pairs of random numbers
    """

    # first, initialize empty arrays of the right size
    x_array = np.empty(n_pairs)
    y_array = np.empty(n_pairs)
    # fill them
    for i in range(n_pairs):
        x_array[i] = np.random.random()
        y_array[i] = np.random.random()
    # plot them
    mpl.pyplot.plot(x_array, y_array, ".")
    mpl.pyplot.show()

def pr6_better(n_pairs):
    """A more pythonic version"""

    # generate a pre-filled random 2d array
    rand = np.random.random((2,n_pairs))
    # plot it!
    mpl.pyplot.plot(rand[0], rand[1], ".")
    mpl.pyplot.show()
```



Output for 5000 points: