EE16A Homework 5

Question 2: Counting The Paths of a Random Surfer

In [47]:
```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

T = np.array([
    [0, 1, 1/3, 1/3],
    [0, 0, 1/3, 1/3],
    [0, 0, 0, 1/3],
    [1, 0, 1/3, 0]
])

eig_val, normalized_eig = np.linalg.eig(T)

print(eig_val)
print()

eig_vec = normalized_eig[:,0]
print("Correponding normalized eig_vec for lambda = 1 is:")
print(eig_vec)
print()

print("Corresponding eig_vec that has values sum to 1 is:")
print(eig_vec / sum(eig_vec))
```

```
[ 1.         +0.j          -0.33333333+0.47140452j -0.33333333-0.47140
452j
 -0.33333333+0.j        ]

Correponding normalized eig_vec for lambda = 1 is:
[-0.61357199+0.j -0.306786  +0.j -0.2300895 +0.j -0.69026849+0.j]

Corresponding eig_vec that has values sum to 1 is:
[0.33333333-0.j 0.16666667-0.j 0.125     -0.j 0.375     -0.j]
```

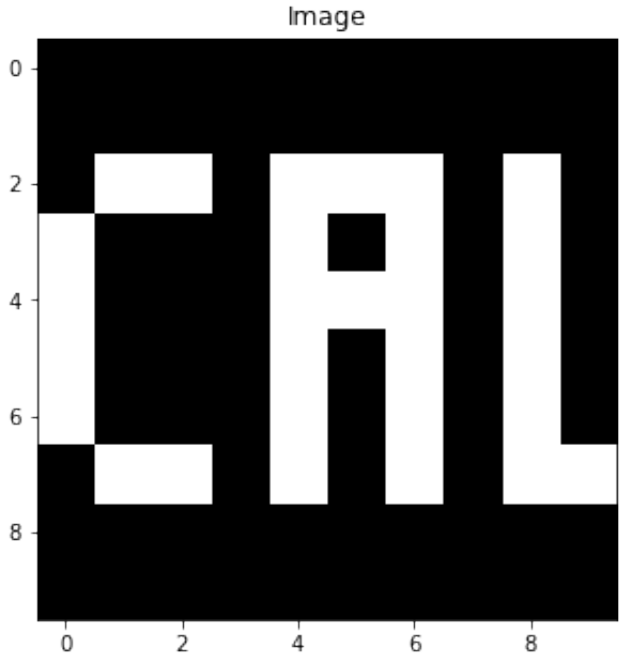
Question 3: Noisy Images

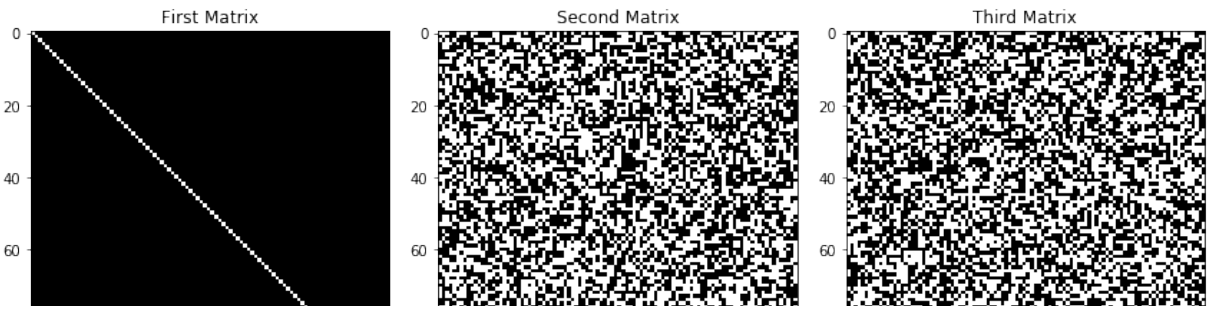
In [48]:
```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

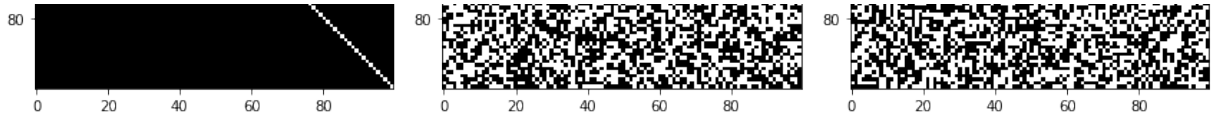
Question 3: Part c

In [50]:
```
# Let's load some data to start off with.
A3 = np.loadtxt("cond_10e6.txt", delimiter=',').reshape(100,100)
A2 = np.loadtxt("cond_1e3.txt", delimiter=',').reshape(100,100)
A1 = np.eye(100)
img = np.loadtxt("image.txt", delimiter=',').reshape(10,10)

# The code below displays the image and the set of masks.
plt.figure(figsize=(5,5))
plt.imshow(img,cmap='gray')
plt.title('Image')
plt.figure(figsize=(12,5))
plt.subplot(131)
plt.imshow(A1,cmap='gray')
plt.title('First Matrix')
plt.subplot(132)
plt.imshow(A2,cmap='gray')
plt.title('Second Matrix')
plt.subplot(133)
plt.imshow(A3,cmap='gray')
plt.title('Third Matrix')
plt.tight_layout()
```







Question 3: Parts d

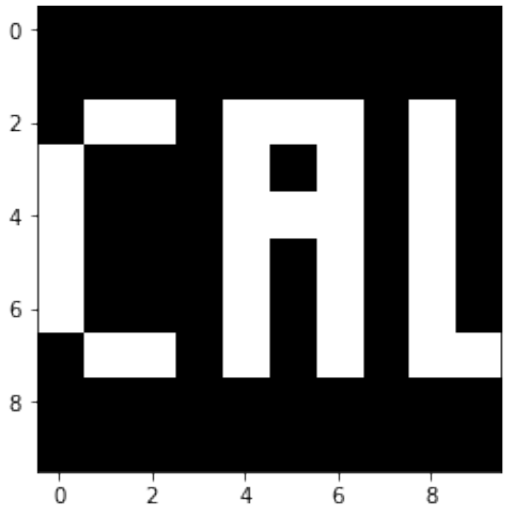
In [51]:
```
# We'll use numpy.random to make some noise.
noise = np.random.normal(0.5,0.1)

# Lets compute the b vector for each matrix and add some noise to the
b1 = A1.dot(img.reshape(100)) + noise
b2 = A2.dot(img.reshape(100)) + noise
b3 = A3.dot(img.reshape(100)) + noise
```

In [52]:
```
# First, let's compute x1 after adding noise and find the minimum eige
x1 = np.linalg.inv(A1).dot(b1)
eigenvalues1 = np.linalg.eig(A1)[0]
print("Is the matrix invertible?", abs(np.linalg.det(A1)) > 0.5)
print("The smallest eigenvalue is:", min(np.absolute(eigenvalues1)))
print("Number of eigenvectors:", len(eigenvalues1))
plt.imshow(x1.reshape(10,10), cmap='gray')
```

```
Is the matrix invertible? True
The smallest eigenvalue is: 1.0
Number of eigenvectors: 100
```

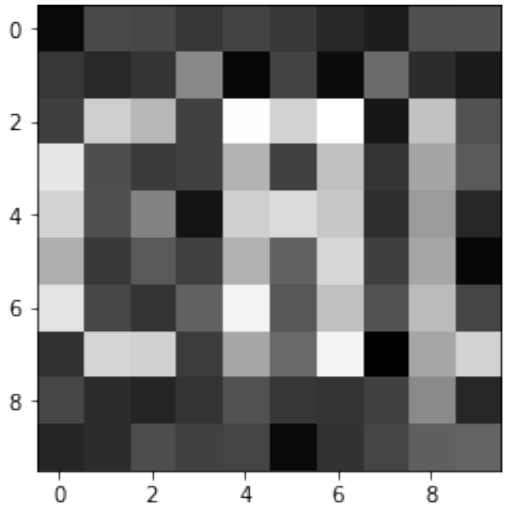
Out[52]:
```
<matplotlib.image.AxesImage at 0x10fe5f5f8>
```



In [53]:
```
# Now let's compute x2 and find the minimum eigenvalue of A2.
x2 = np.linalg.inv(A2).dot(b2)
eigenvalues2 = np.linalg.eig(A2)[0]
print("Is the matrix invertible?", abs(np.linalg.det(A2)) > 0.5)
print("The smallest eigenvalue is:", min(np.absolute(eigenvalues2)))
print("Number of eigenvectors:", len(eigenvalues2))
plt.imshow(x2.reshape(10,10), cmap='gray')
```

```
Is the matrix invertible? True
The smallest eigenvalue is: 0.29516363308630184
Number of eigenvectors: 100
```

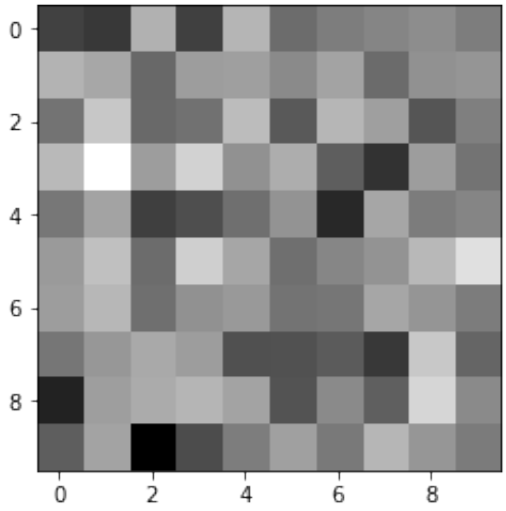
Out[53]: <matplotlib.image.AxesImage at 0x119db5828>



In [54]:
```
# Now let's compute x3 and find the minimum eigenvalue of A3.
x3 = np.linalg.inv(A3).dot(b3)
eigenvalues3 = np.linalg.eig(A3)[0]
print("Is the matrix invertible?", abs(np.linalg.det(A3)) > 0.5)
print("The smallest eigenvalue is:", min(np.absolute(eigenvalues3)))
print("Number of eigenvectors:", len(eigenvalues3))
plt.imshow(x3.reshape(10,10), cmap='gray')
```

```
Is the matrix invertible? True
The smallest eigenvalue is: 1.2184217510026823e-05
Number of eigenvectors: 100
```

Out[54]: <matplotlib.image.AxesImage at 0x11a624cc0>



In [7]:

In [ ]: