In [1]:

```
# For scientific computing
import numpy as np

# For plotting
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_context('notebook')
%matplotlib inline
```

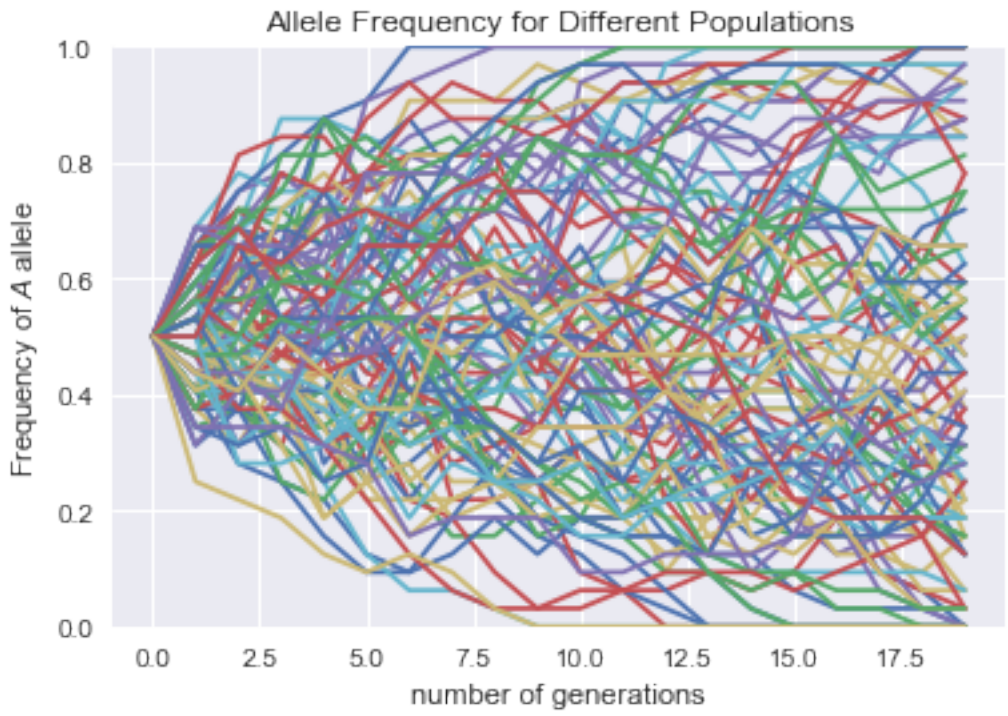
Question 1c

In [2]:

```
# Define some parameters of the simulation.
pop_size = 16  # Number of individuals in the population.
n_alleles = 2 * pop_size  # Total number of alleles in the population.
n_gen = 20 # Number of generations to track the allele frequencies.
freq = 0.5  # Allele frequency of allele A.
n_populations = 107
frequency = np.zeros((n_populations, n_gen))

# Loop through each herd and repeat the simulation.
for i in range(n_populations):
    # Set the initial frequency.
    frequency[i, 0] = 0.5
    for j in range(1, n_gen):
        # Flip the coins.
        flips = np.random.rand(n_alleles)

        # Find the number of 'A' alleles.
        num_A = np.sum(flips < frequency[i, j-1])

        # Determine the new frequency
        new_freq = num_A / n_alleles

        # Set the current allele frequency.
        frequency[i, j] = new_freq
# Plot the trajectories from all of the herds.
gen_vec = range(0,n_gen,1)
for i in range(n_populations):
    plt.plot(gen_vec, frequency[i, :])

# Add axis labels.
plt.xlabel('number of generations')
plt.ylabel('Frequency of $A$ allele')
plt.title("Allele Frequency for Different Populations")
plt.ylim([0, 1])
```

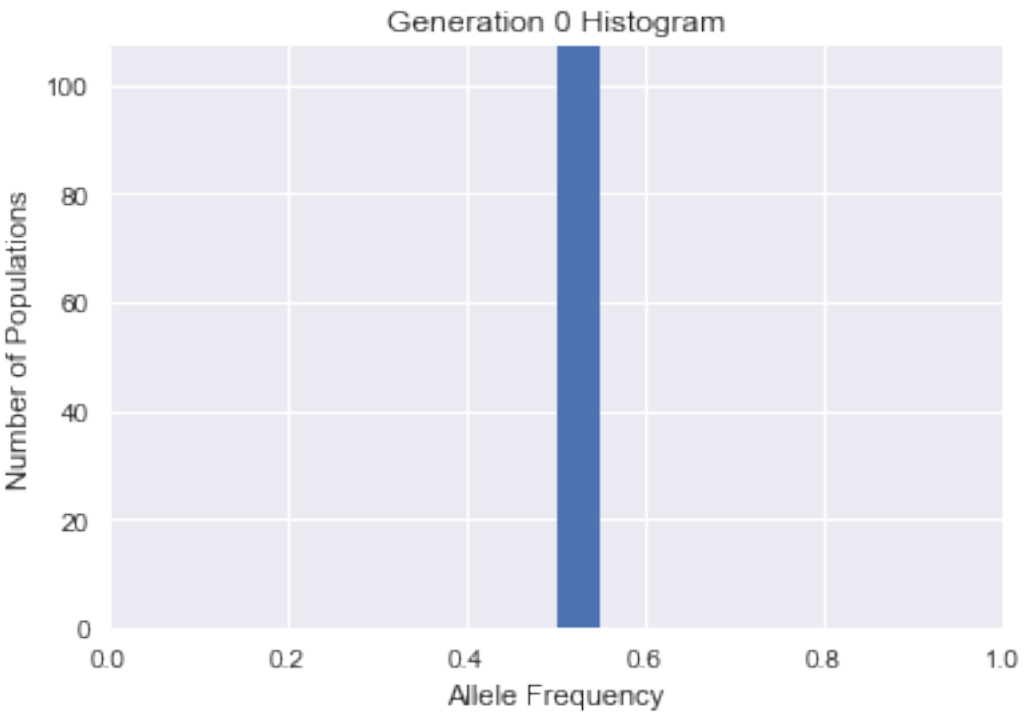
Out[2]:

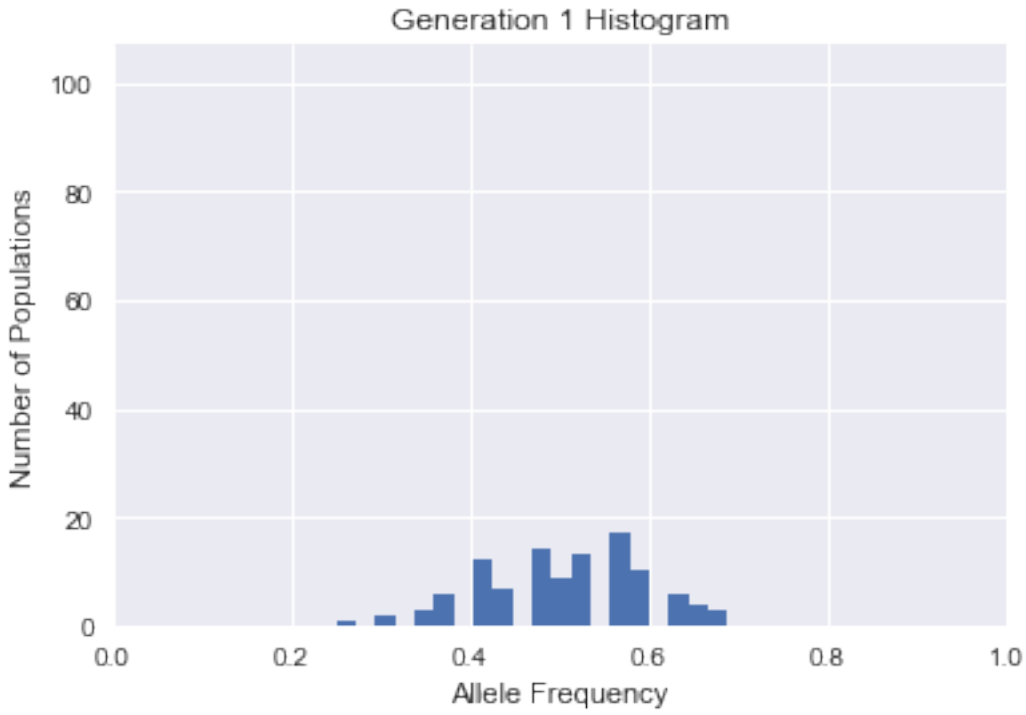
(0, 1)

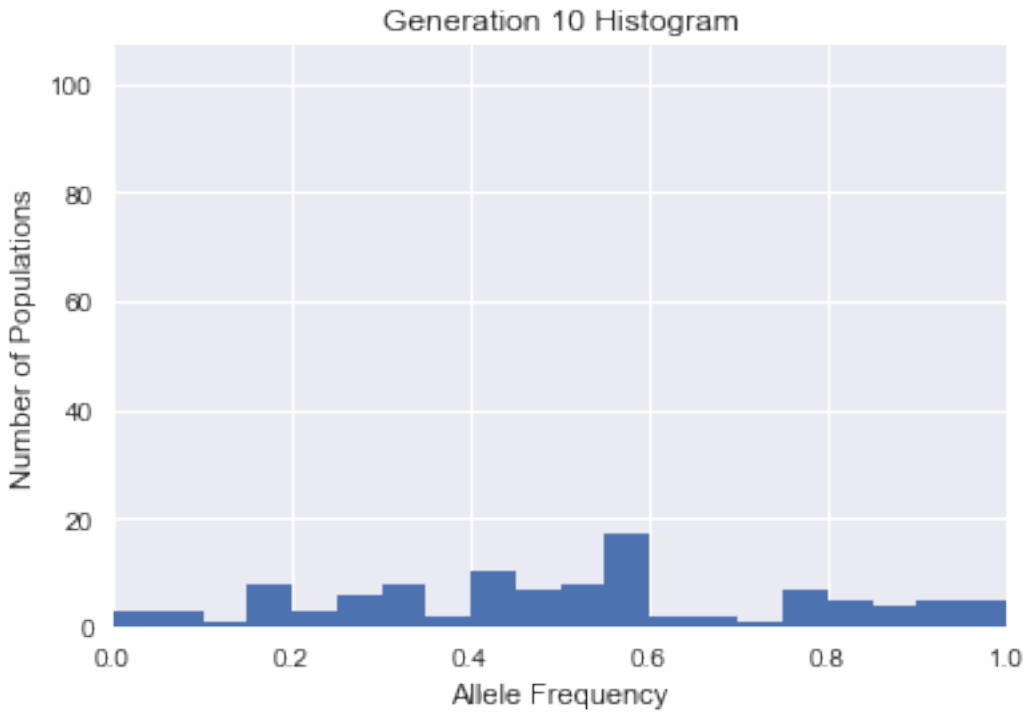


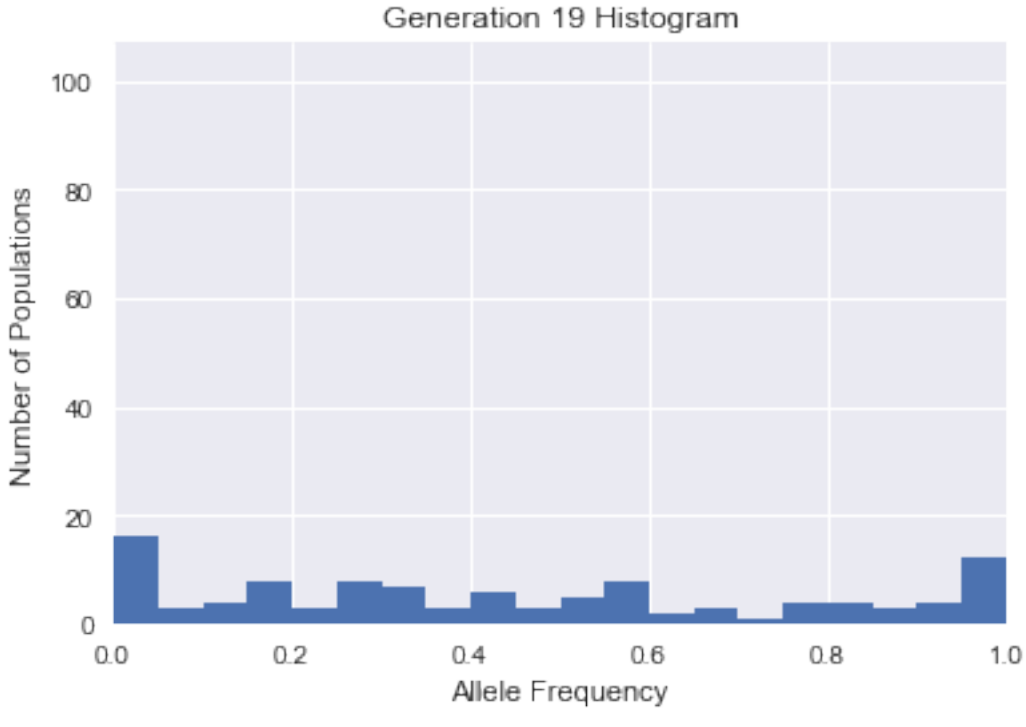
In [3]:

```
histVector = [0,1,10,19]
for hist in histVector:
    plt.figure(hist)
    plt.hist(frequency[:, hist], bins=20)
    plt.xlabel('Allele Frequency')
    plt.ylabel('Number of Populations')
    plt.xlim([0,1])
    plt.ylim([0,107])
    plt.title("Generation " + str(hist) + " Histogram")
```




Generation 1 Histogram
Number of Populations
0
20
40
60
80
100
0.0
0.2
0.4
0.6
0.8
1.0
Allele Frequency


Generation 10 Histogram
Number of Populations
0
20
40
60
80
100
0.0
0.2
0.4
0.6
0.8
1.0
Allele Frequency

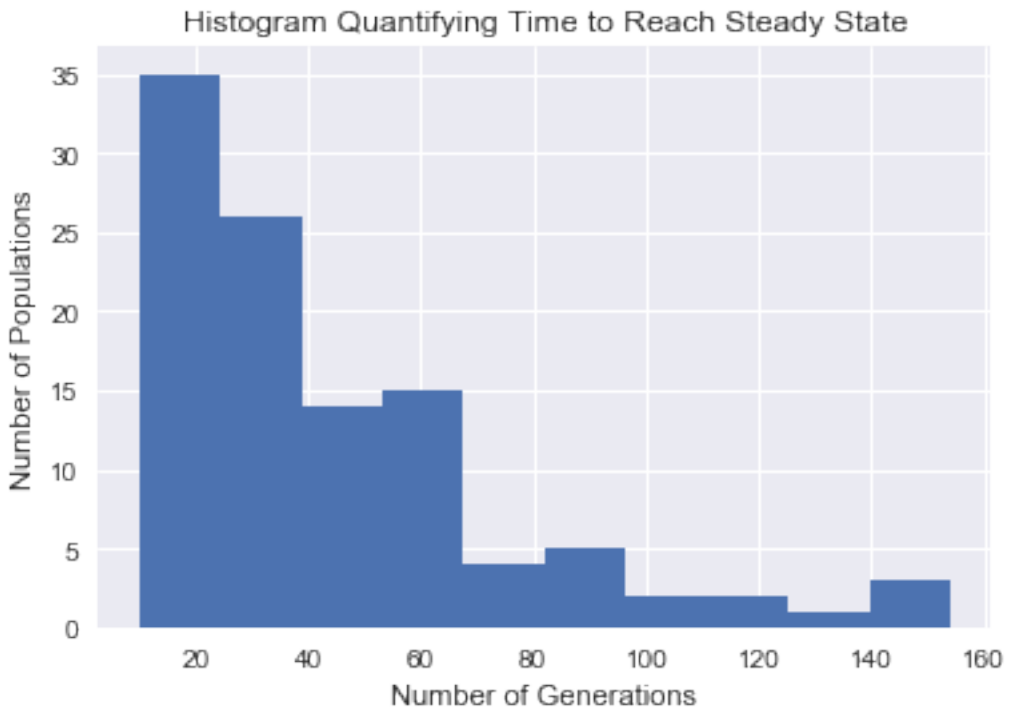

Generation 19 Histogram
Number of Populations
0
20
40
60
80
100
0.0
0.2
0.4
0.6
0.8
1.0
Allele Frequency

Question 1d

In [4]:

```
pop_size = 16  # Number of individuals in the population.
n_alleles = 2 * pop_size  # Total number of alleles in the population.
n_gen = 1000 # Number of generations to track the allele frequencies.
freq = 0.5  # Allele frequency of allele A.
n_populations = 107
frequency = np.zeros((n_populations, n_gen))
whereBecameZeroOne = np.zeros(n_populations)
# Loop through each herd and repeat the simulation.
for i in range(n_populations):
    # Set the initial frequency.
    frequency[i, 0] = 0.5
    for j in range(1, n_gen):
        # Flip the coins.
        flips = np.random.rand(n_alleles)

        # Find the number of 'A' alleles.
        num_A = np.sum(flips < frequency[i, j-1])

        # Determine the new frequency
        new_freq = num_A / n_alleles

        # Set the current allele frequency.
        frequency[i, j] = new_freq
        if (new_freq==1 or new_freq==0):
            #print("Generation where " + "population "+ str(i+1)+ " fixated is "
+ str(j))
            whereBecameZeroOne[i]=j
            break
gen_vec = np.arange(0, n_gen, 1)
plt.hist(whereBecameZeroOne)
plt.title("Histogram Quantifying Time to Reach Steady State")
plt.xlabel("Number of Generations")
plt.ylabel("Number of Populations")
print("The mean number of generations required was " + str(whereBecameZeroOne.me
an()))
```

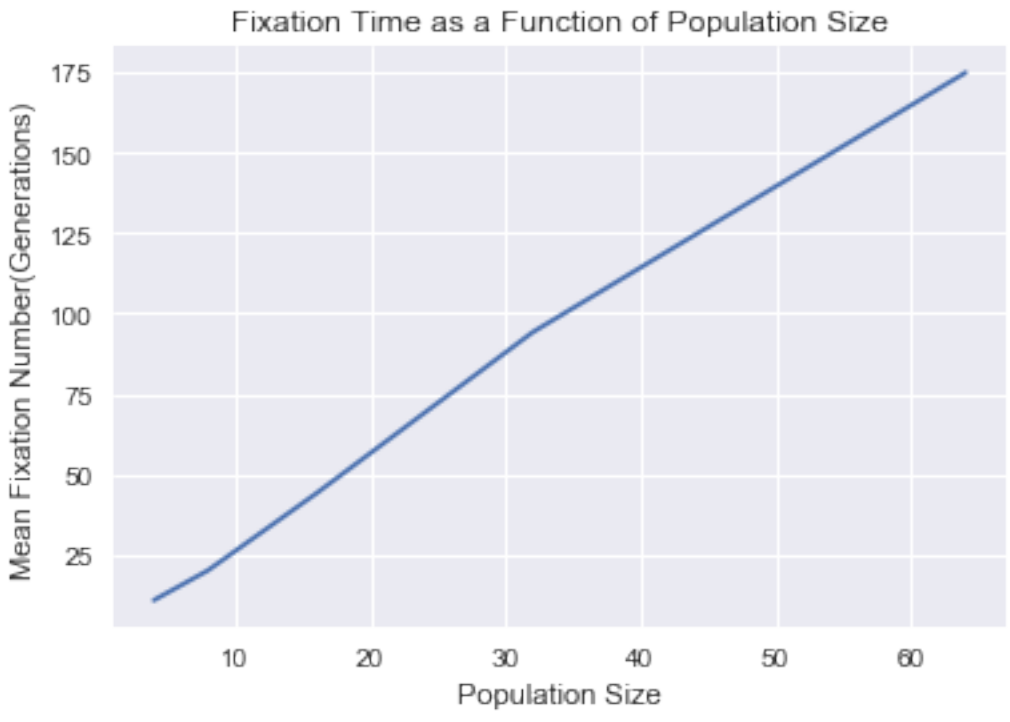
```
The mean number of generations required was 43.9439252336
```



In [5]:

```
populationSize = [4,8,16,32,64]
meanFixation = np.zeros(5)
n_gen = 1000 # Number of generations to track the allele frequencies.
freq = 0.5  # Allele frequency of allele A.
n_populations = 107
for x in range(len(populationSize)):
    n_alleles = 2 * populationSize[x]
    frequency = np.zeros((n_populations, n_gen))
    whereBecameZeroOne = np.zeros(n_populations)
    for i in range(n_populations):
        # Set the initial frequency.
        frequency[i, 0] = 0.5
        for j in range(1, n_gen):
            # Flip the coins.
            flips = np.random.rand(n_alleles)

            # Find the number of 'A' alleles.
            num_A = np.sum(flips < frequency[i, j-1])

            # Determine the new frequency
            new_freq = num_A / n_alleles

            # Set the current allele frequency.
            frequency[i, j] = new_freq
            if (new_freq==1 or new_freq==0):
                whereBecameZeroOne[i]=j
                break
    meanFixation[x]=whereBecameZeroOne.mean()
plt.plot(populationSize, meanFixation)
plt.xlabel("Population Size")
plt.title("Fixation Time as a Function of Population Size")
plt.ylabel("Mean Fixation Number(Generations)")
```

```
Out[5]:

<matplotlib.text.Text at 0x11f4e96d8>
```

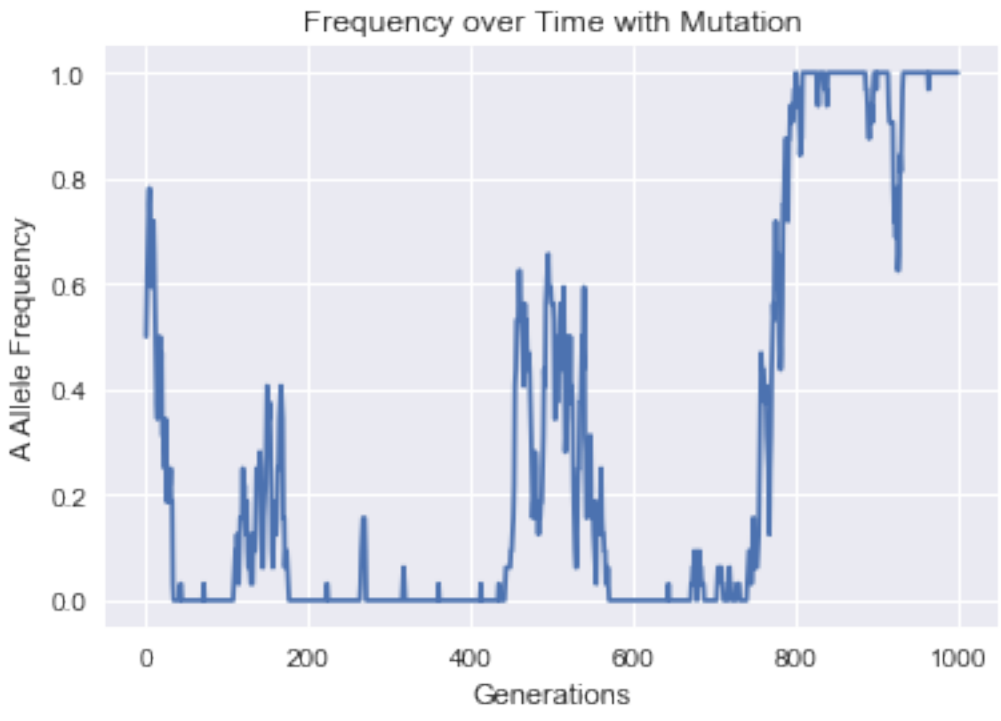


Question 1e

In [6]:

```
# Define some parameters of the simulation.
pop_size = 16  # Number of individuals in the population.
n_alleles = 2 * pop_size  # Total number of alleles in the population.
n_gen = 1000 # Number of generations to track the allele frequencies.
freq = 0.5  # Allele frequency of allele A.

# Set up a vector to store the allele frequency at each generation.
frequency = np.zeros(n_gen)  # Full of zeros.

# Set the initial condition.
frequency[0] = freq

# Loop through the generations.
for i in range(1, n_gen):
    # Flip a coin for each member of the population.
    flips = np.random.rand(n_alleles)
    # Determine if each allele is 'A' or 'a' based on the frequency from
    # the previous generation.
    num_A = np.sum(flips < frequency[i-1])
    num_a = n_alleles-num_A
    finalA = 0
    for x in range(num_A):
        if np.random.rand() > .001:
            finalA+=1
    for x in range(num_a):
        if np.random.rand() < .001:
            finalA+=1
    # Determine the new allele frequency.
    new_freq = finalA / n_alleles

    # Store the new allele frequency.
    frequency[i] = new_freq
    # Set up a vector for the number of generations.
gen_vec = np.arange(0, n_gen, 1)

# Plot this single trajectory.
plt.plot(gen_vec, frequency)
plt.title("Frequency over Time with Mutation")
plt.xlabel("Generations")
plt.ylabel("A Allele Frequency")
```

```
Out[6]:

<matplotlib.text.Text at 0x11f10a390>
```



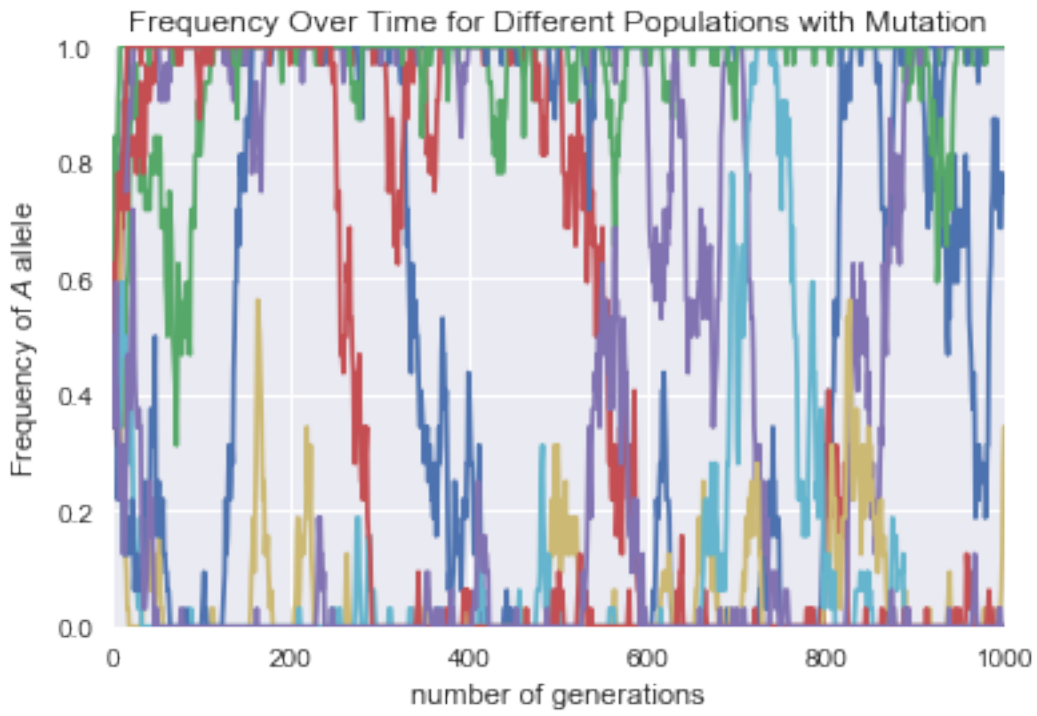
Question 1f

In [7]:

```
# Define some parameters of the simulation.
pop_size = 16  # Number of individuals in the population.
n_alleles = 2 * pop_size  # Total number of alleles in the population.
n_gen = 1000 # Number of generations to track the allele frequencies.
freq = 0.5  # Allele frequency of allele A.
n_populations = 100
frequency = np.zeros((n_populations, n_gen))

# Loop through each herd and repeat the simulation.
for i in range(n_populations):
    # Set the initial frequency.
    frequency[i, 0] = 0.5
    for j in range(1, n_gen):
        # Flip the coins.
        flips = np.random.rand(n_alleles)

        # Find the number of 'A' alleles.
        ### NEW CODE
        num_A = np.sum(flips < frequency[i, j-1])
        num_a = n_alleles-num_A
        finalA = 0
        for x in range(num_A):
            if np.random.rand() > .001:
                finalA+=1
        for x in range(num_a):
            if np.random.rand() < .001:
                finalA+=1
        # Determine the new allele frequency.
        new_freq = finalA / n_alleles

        # Set the current allele frequency.
        frequency[i, j] = new_freq
# Plot the trajectories from all of the herds.
for i in range(0,n_populations,10):
    plt.plot(gen_vec, frequency[i, :])

# Add axis labels.
plt.title("Frequency Over Time for Different Populations with Mutation")
plt.xlabel('number of generations')
plt.ylabel('Frequency of $A$ allele')
plt.ylim([0, 1])
plt.xlim([0,1000])
```

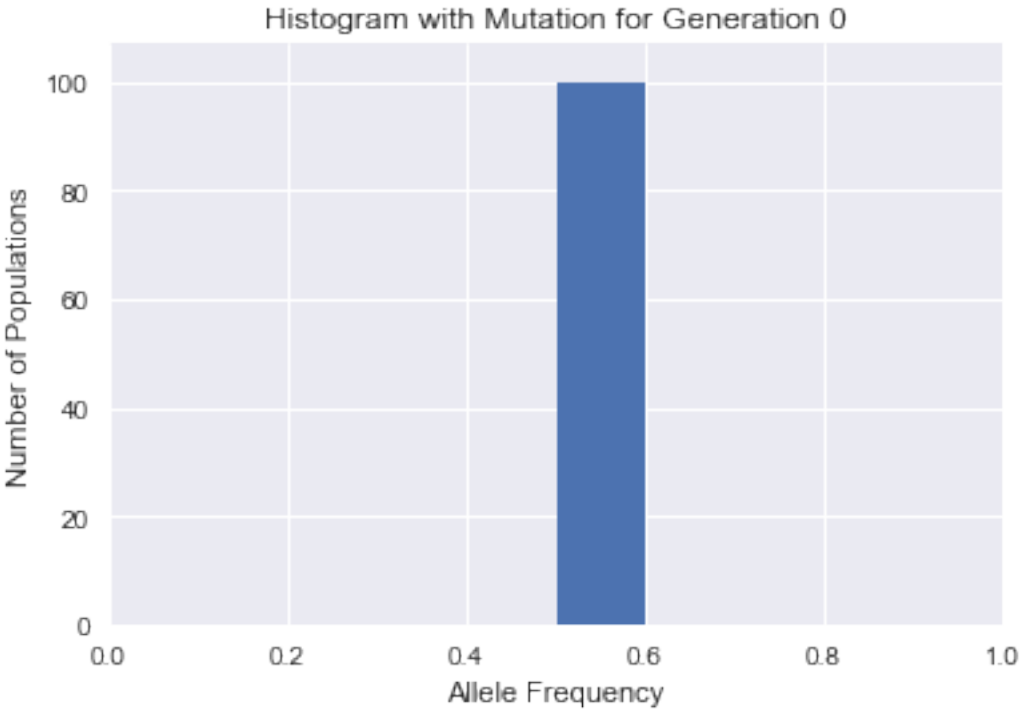
Out[7]:

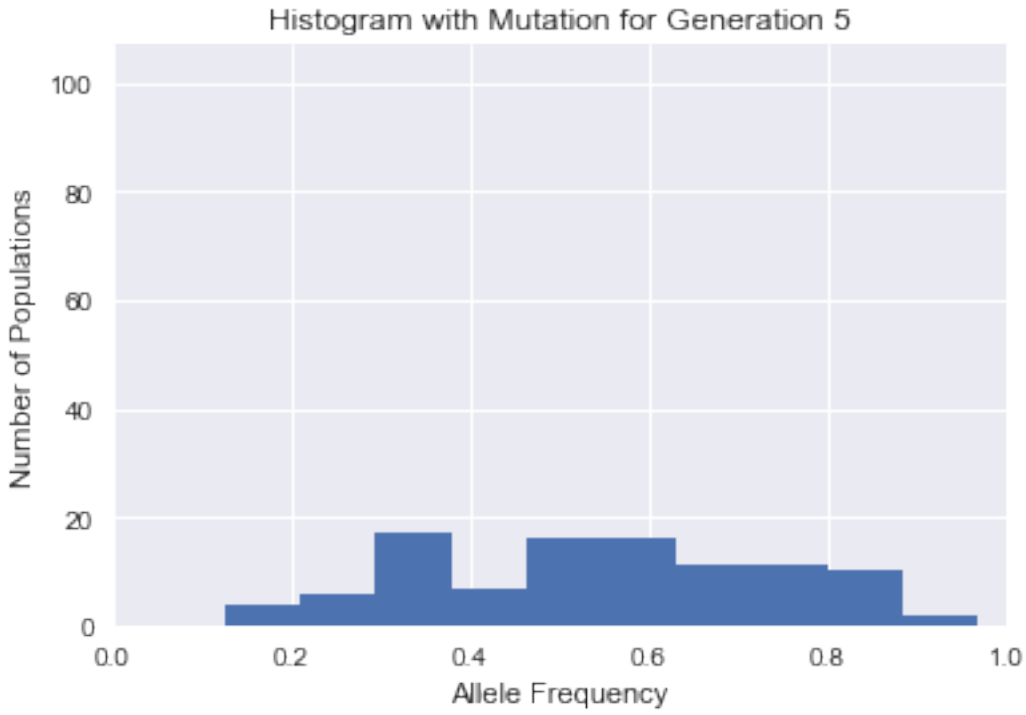
(0, 1000)

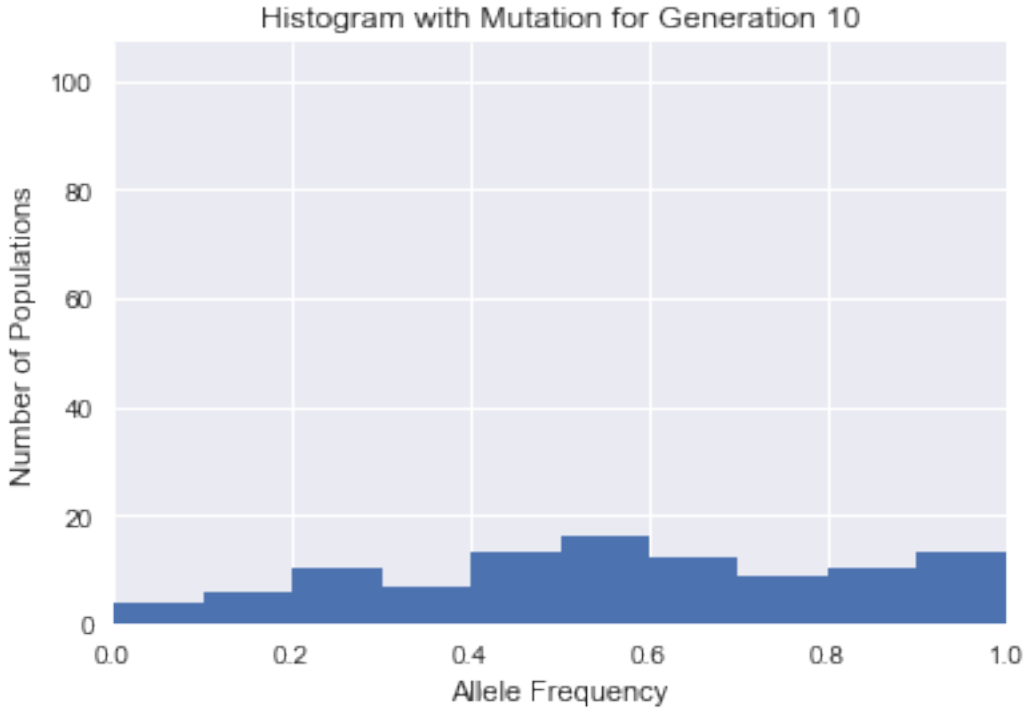


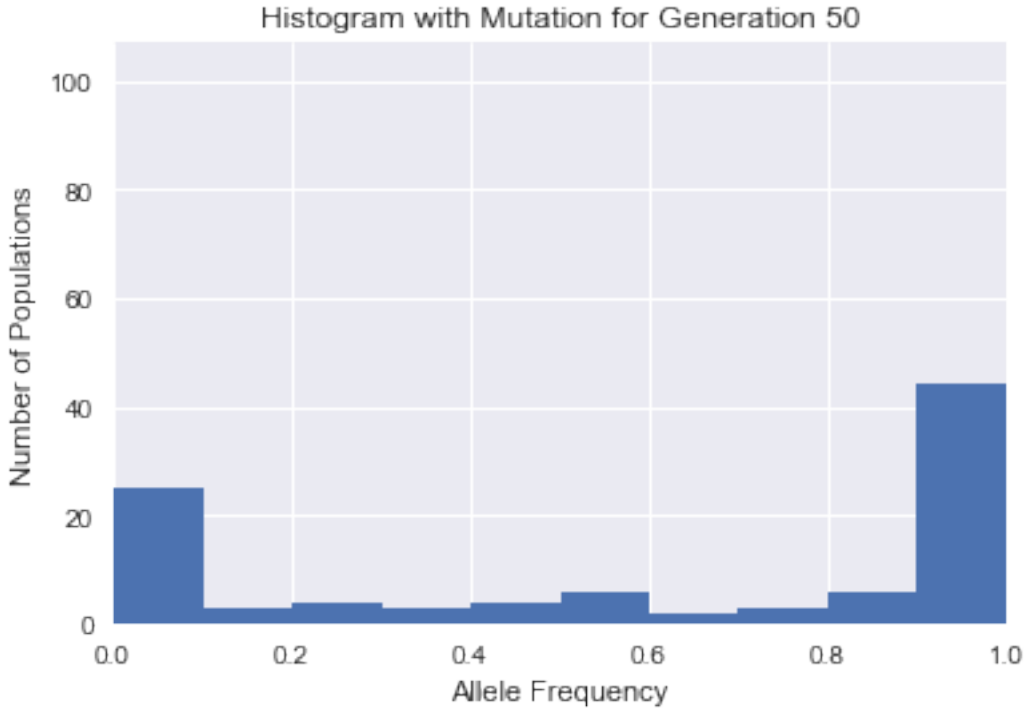
In [221]:

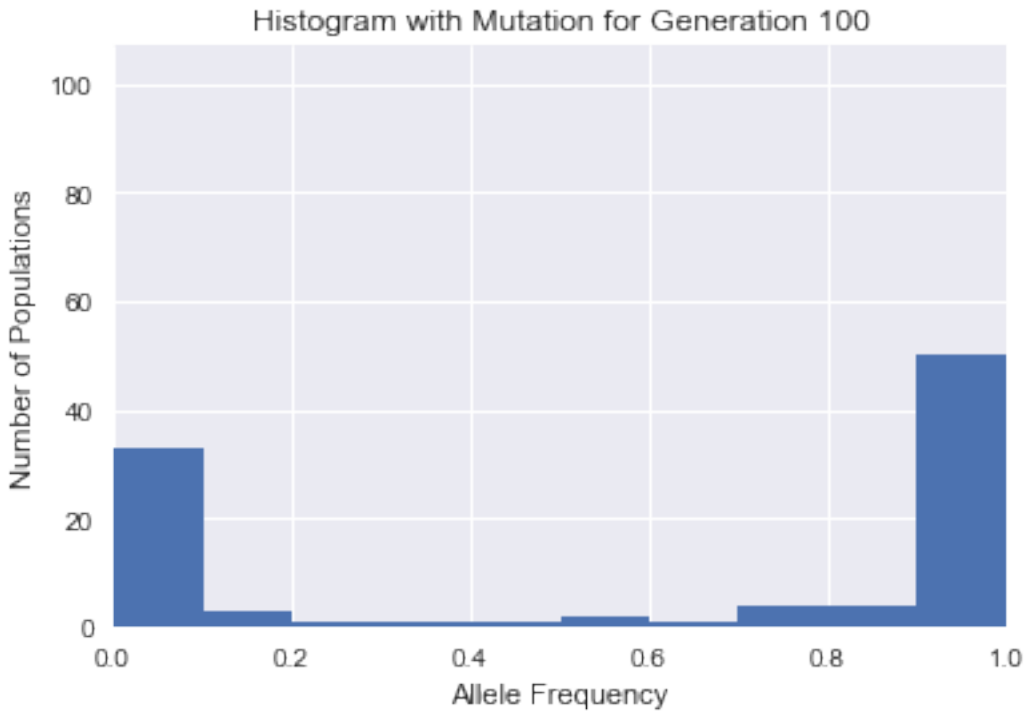
```
histVector = [0,5,10,50,100,500]
for x in histVector:
    plt.figure(x)
    plt.hist(frequency[:, x])
    plt.xlabel('Allele Frequency')
    plt.ylabel('Number of Populations')
    plt.xlim([0,1])
    plt.ylim([0,107])
    plt.title("Histogram with Mutation for Generation "+str(x))
```

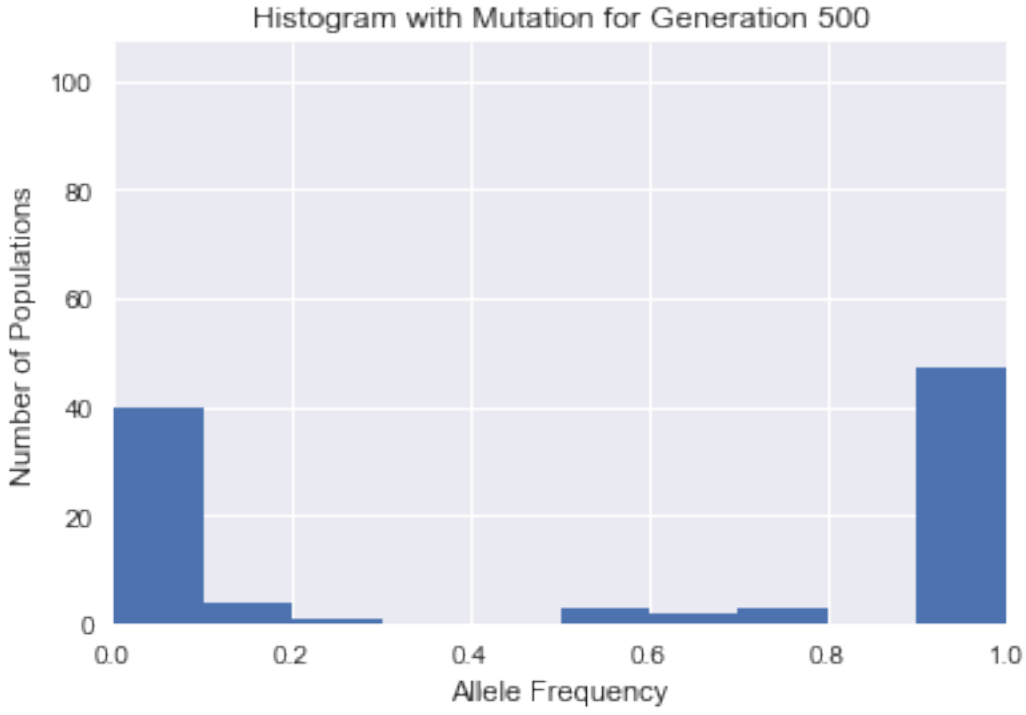



Histogram with Mutation for Generation 5
Number of Populations
0
20
40
60
80
100
0.0
0.2
0.4
0.6
0.8
1.0
Allele Frequency


Histogram with Mutation for Generation 10
ber of Populations
0
20
40
60
80
100
0.0
0.2
0.4
0.6
0.8
1.0
Allele Frequency


Histogram with Mutation for Generation 50
Number of Populations
0
20
40
60
80
100
0.0
0.2
0.4
0.6
0.8
1.0
Allele Frequency


Histogram with Mutation for Generation 100
Number of Populations
0
20
40
60
80
100
0.0
0.2
0.4
0.6
0.8
1.0
Allele Frequency

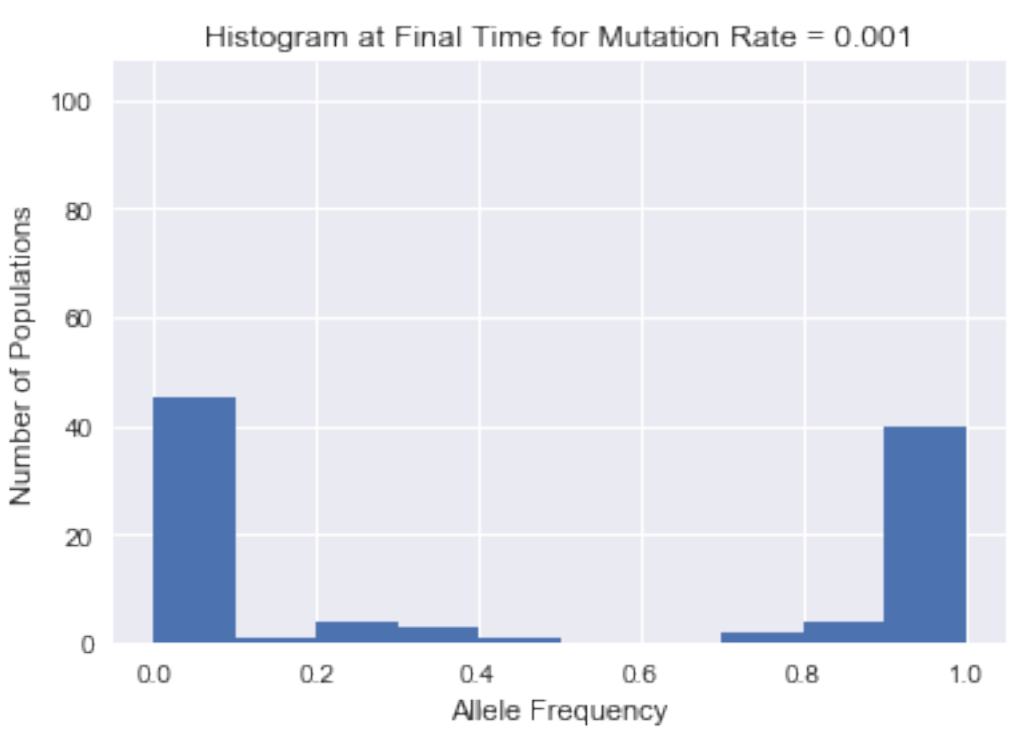

Histogram with Mutation for Generation 500
Number of Populations
0
20
40
60
80
100
0.0
0.2
0.4
0.6
0.8
1.0
Allele Frequency

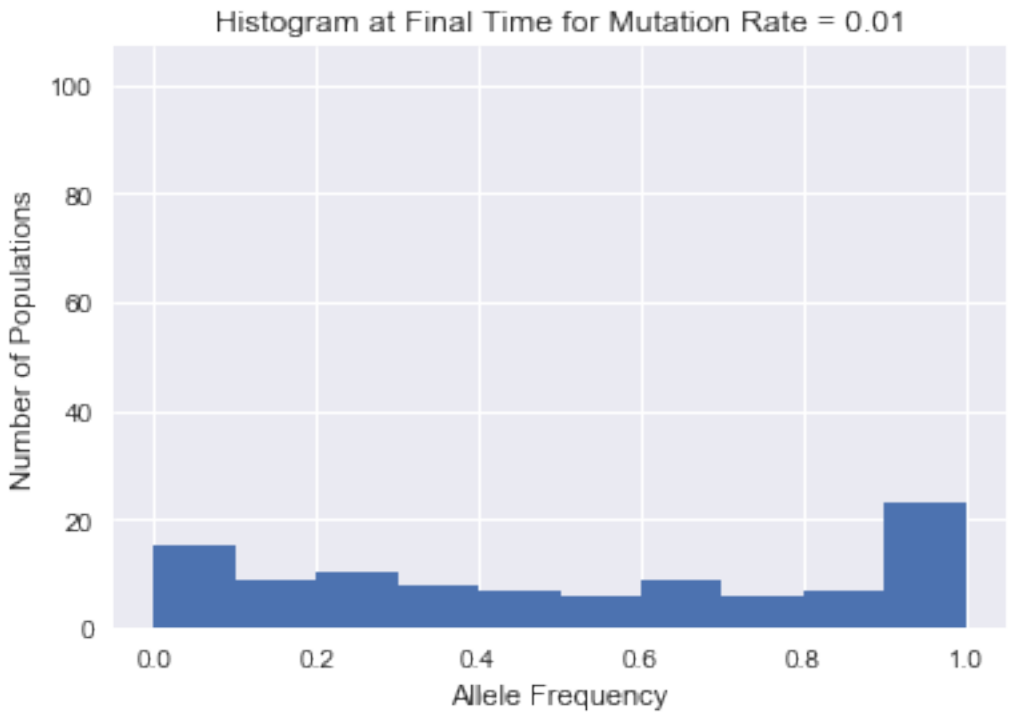
Question 1g

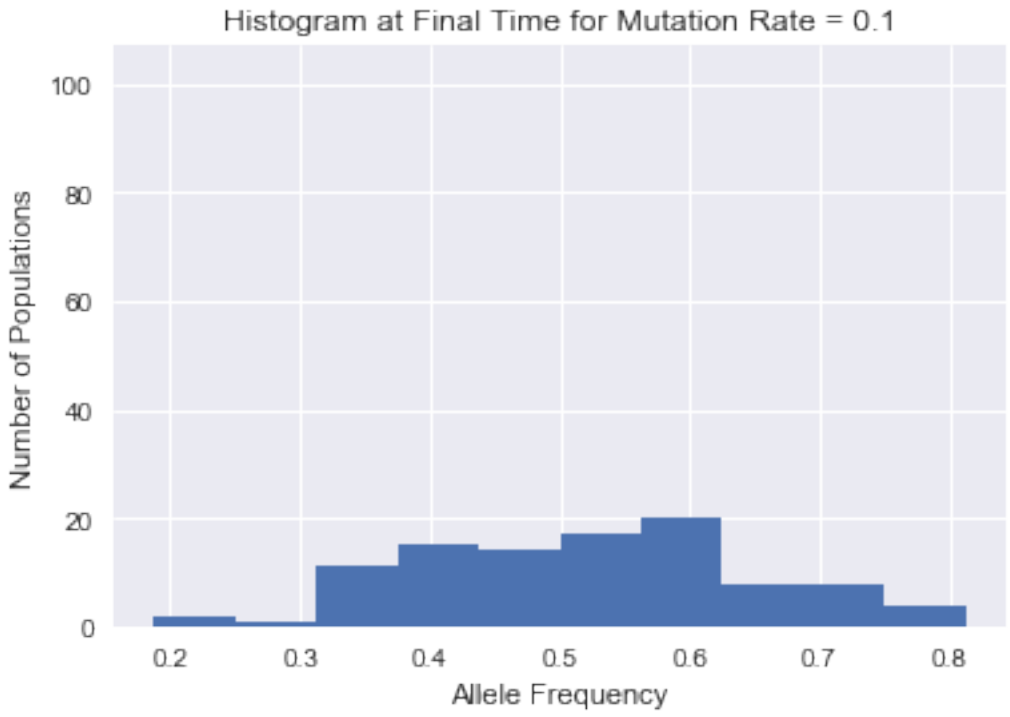
In [8]:

```
mutations = [0,.001,.01,.1]
pop_size = 16  # Number of individuals in the population.
n_alleles = 2 * pop_size  # Total number of alleles in the population.
n_gen = 1000 # Number of generations to track the allele frequencies.
freq = 0.5  # Allele frequency of allele A.
n_populations = 100
for mutation in range(len(mutations)):
    frequency = np.zeros((n_populations, n_gen))
    for i in range(n_populations):
        # Set the initial frequency.
        frequency[i, 0] = 0.5
        for j in range(1, n_gen):
            # Flip the coins.
            flips = np.random.rand(n_alleles)

            # Find the number of 'A' alleles.
            num_A = np.sum(flips < frequency[i, j-1])

            num_a = n_alleles-num_A
            finalA = 0
            for x in range(num_A):
                if np.random.rand() > mutations[mutation]:
                    finalA+=1
            for x in range(num_a):
                if np.random.rand() < mutations[mutation]:
                    finalA+=1
            # Determine the new allele frequency.
            new_freq = finalA / n_alleles

            # Set the current allele frequency.
            frequency[i, j] = new_freq
    plt.figure(mutation+1)
    plt.hist(frequency[:,999])
    plt.ylim([0,107])
    plt.title("Histogram at Final Time for Mutation Rate = " + str(mutations[mut
ation]))
    plt.xlabel("Allele Frequency")
    plt.ylabel("Number of Populations")
```


Histogram at Final Time for Mutation Rate = 0
Number of Populations
0
20
40
60
80
100
0.0
0.2
0.4
0.6
0.8
1.0
Allele Frequency


Histogram at Final Time for Mutation Rate = 0.001
Number of Populations
0
20
40
60
80
100
0.0
0.2
0.4
0.6
0.8
1.0
Allele Frequency


Histogram at Final Time for Mutation Rate = 0.01
Number of Populations
0
20
40
60
80
100
0.0
0.2
0.4
0.6
0.8
1.0
Allele Frequency

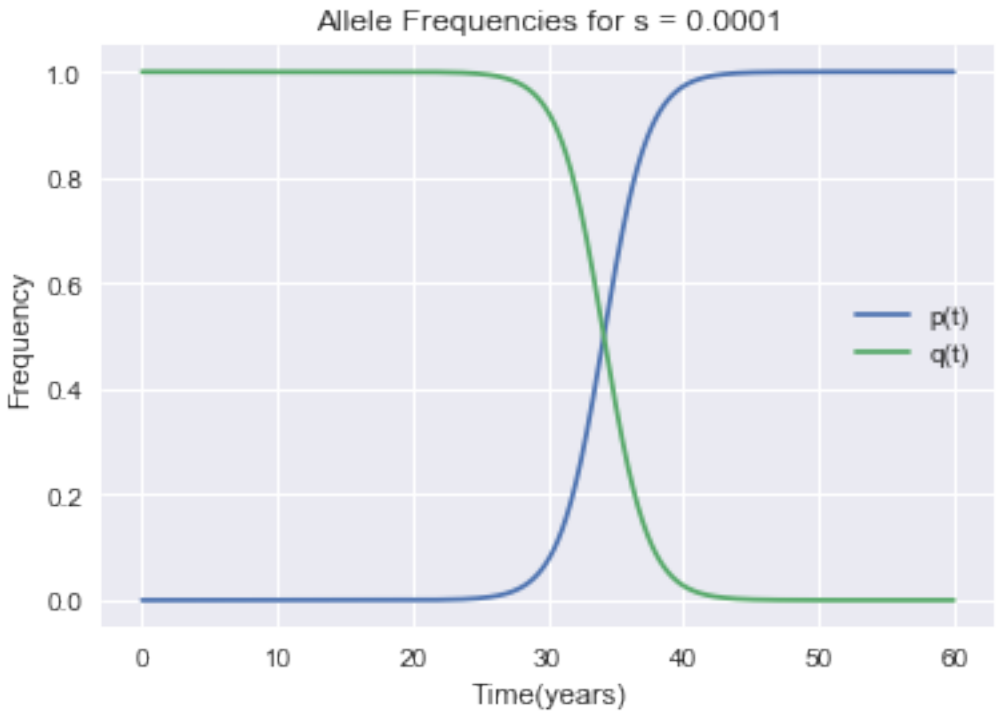


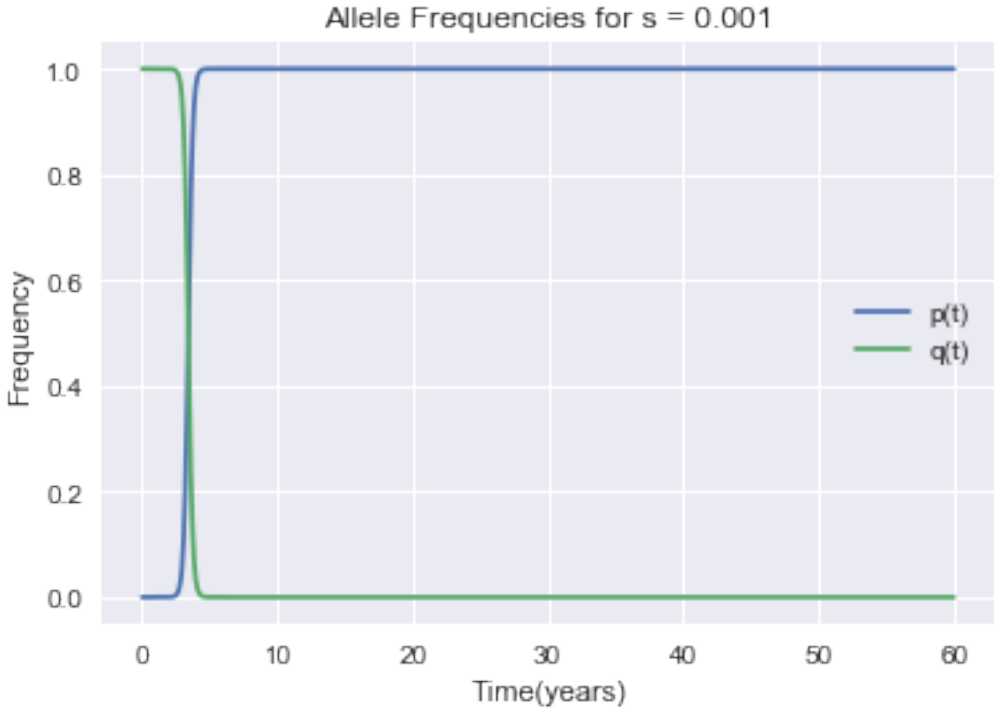
Question 2c

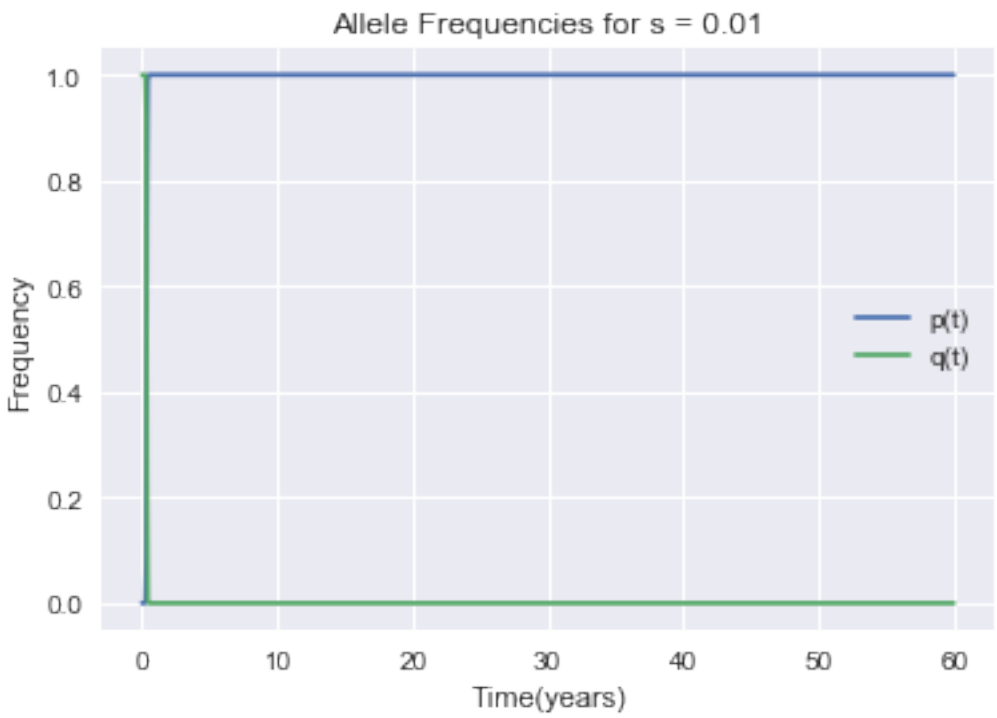
In [9]:

```
rates = [10**-4, 10**-3,10**-2]
for num in range(len(rates)):
    x = np.linspace(0,60,1000)
    y1 = 1/(1+10**(9)* np.exp(-rates[num]*np.log(2)*8760*x))
    y2 = 1/(1+10**(-9)* np.exp(rates[num]*np.log(2)*8760*x))
    plt.figure(num)
    plt.title("Allele Frequencies for s = "+ str(rates[num]))
    plt.xlabel("Time(years)")
    plt.ylabel("Frequency")
    plt.plot(x, y1)
    plt.plot(x, y2)
    plt.legend(["p(t)","q(t)"])
```

```
/Users/harryputterman/anaconda/lib/python3.6/site-packages/ipykernel
/__main__.py:5: RuntimeWarning: overflow encountered in exp
```


Allele Frequencies for s = 0.0001
Frequency
1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40
50
60
Time(years)
p(t)
q(t)


Allele Frequencies for s = 0.001
Frequency
1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40
50
60
Time(years)
p(t)
q(t)


Allele Frequencies for s = 0.01
Frequency
1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40
50
60
Time(years)
p(t)
q(t)

In [ ]: