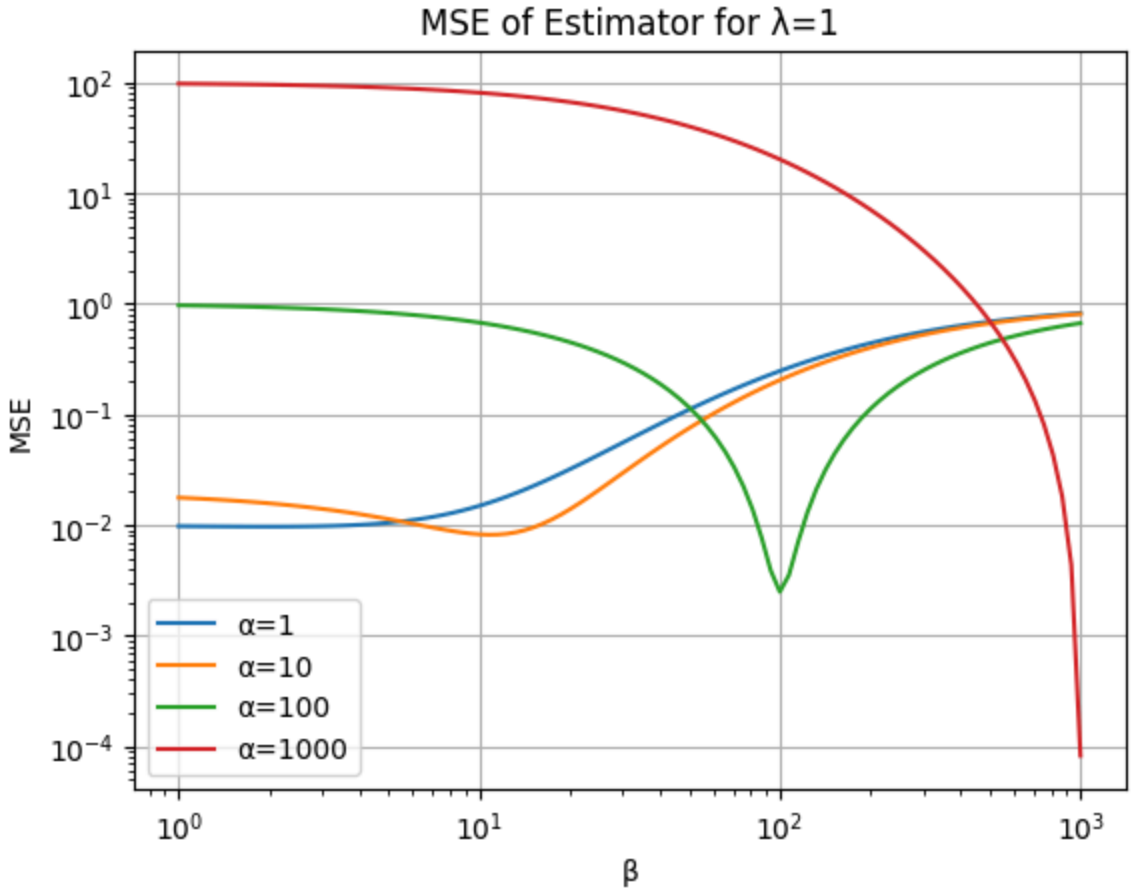
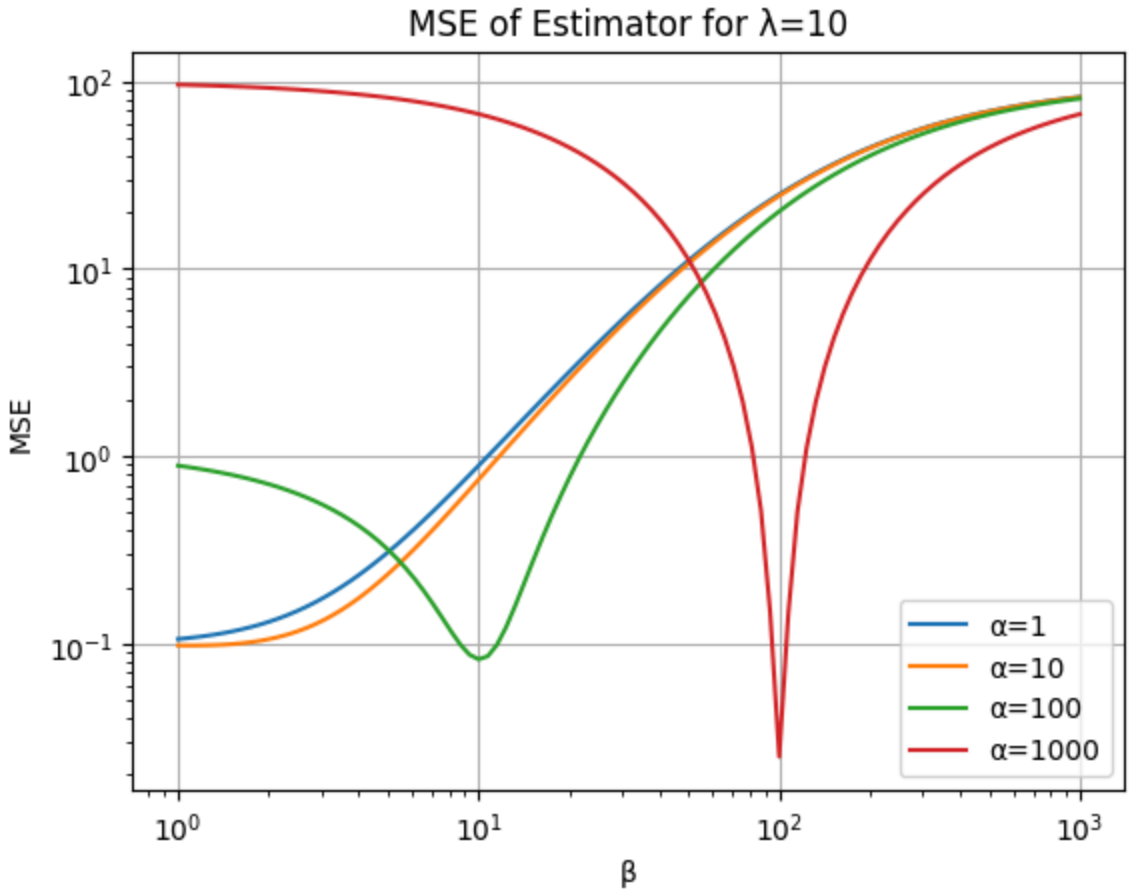
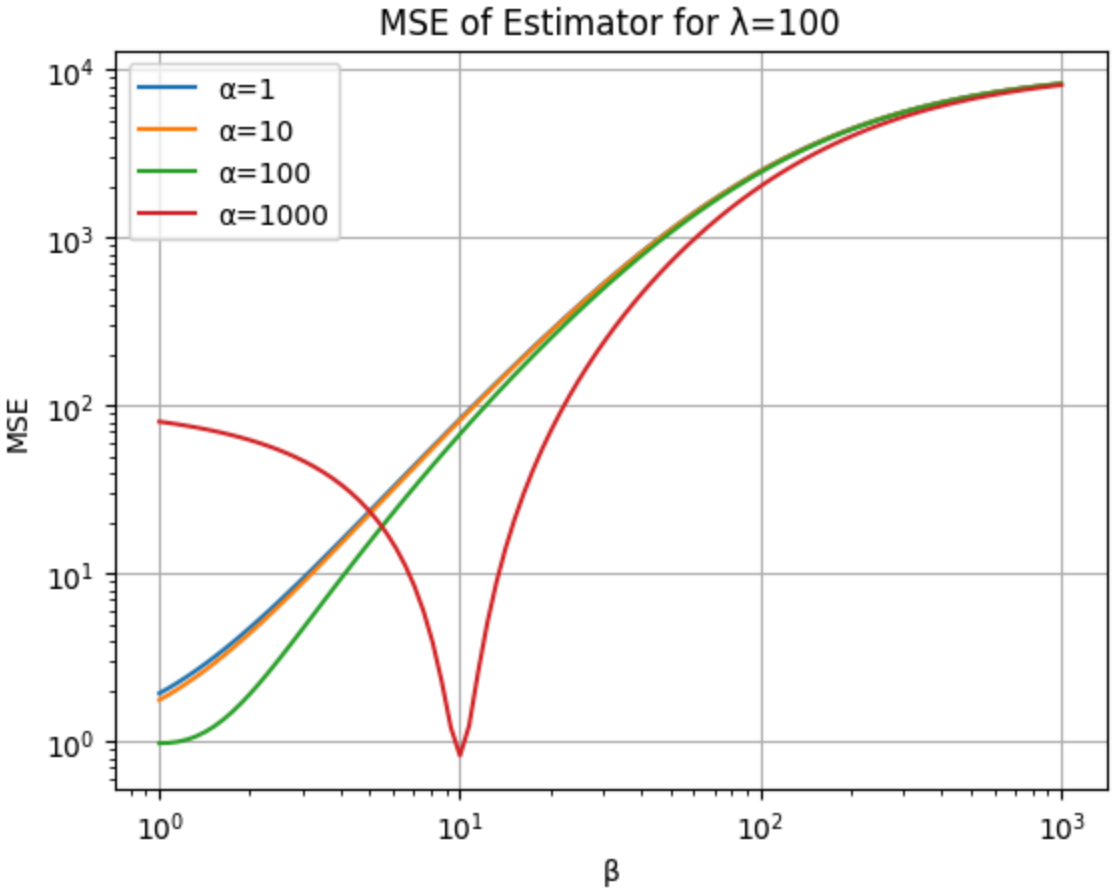
2

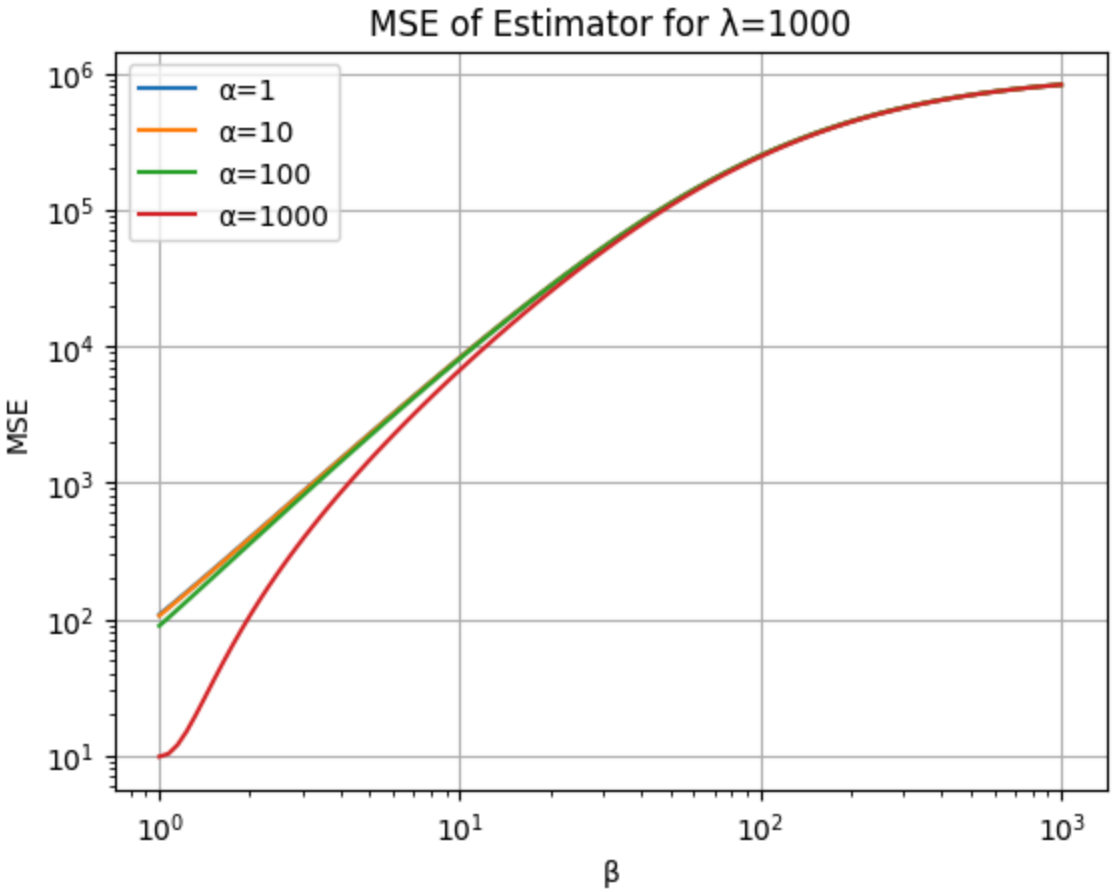
i

In [16]:
```
import numpy as np
import matplotlib.pyplot as plt

lambdas = [1, 10, 100, 1000]
n = 100

beta_values = np.logspace(0, 3, 100)
alpha_values = [1, 10, 100, 1000]

for l in lambdas:
    plt.figure()
    for a in alpha_values:
        mse_values = []
        for b in beta_values:
            # MSE from P1 iii
            mse = (n * l + (a - l * b) ** 2) / (n + b) ** 2
            mse_values.append(mse)
        plt.plot(beta_values, mse_values, label=f'α={a}')
    plt.xscale('log')
    plt.yscale('log')
    plt.xlabel('β')
    plt.ylabel('MSE')
    plt.title(f'MSE of Estimator for λ={l}')
    plt.legend()
    plt.grid(True)
    plt.show()
```


MSE of Estimator for λ=1
MSE
β
α=1
α=10
α=100
α=1000


MSE of Estimator for λ=10
MSE
β
α=1
α=10
α=100
α=1000


MSE of Estimator for λ=100
α=1
α=10
α=100
α=1000
MSE
β
10^0
10^1
10^2
10^3
10^4


MSE of Estimator for λ=1000
α=1
α=10
α=100
α=1000
MSE
β
10^0
10^1
10^2
10^3
10^4
10^5
10^6

ii

For lower λ values, there are clear local minima for the MSE of given certain α, β values. However, as λ increases, the MSE also increases exponenentialy with no local minima apart from larger and larger α values as β also increases.

iii

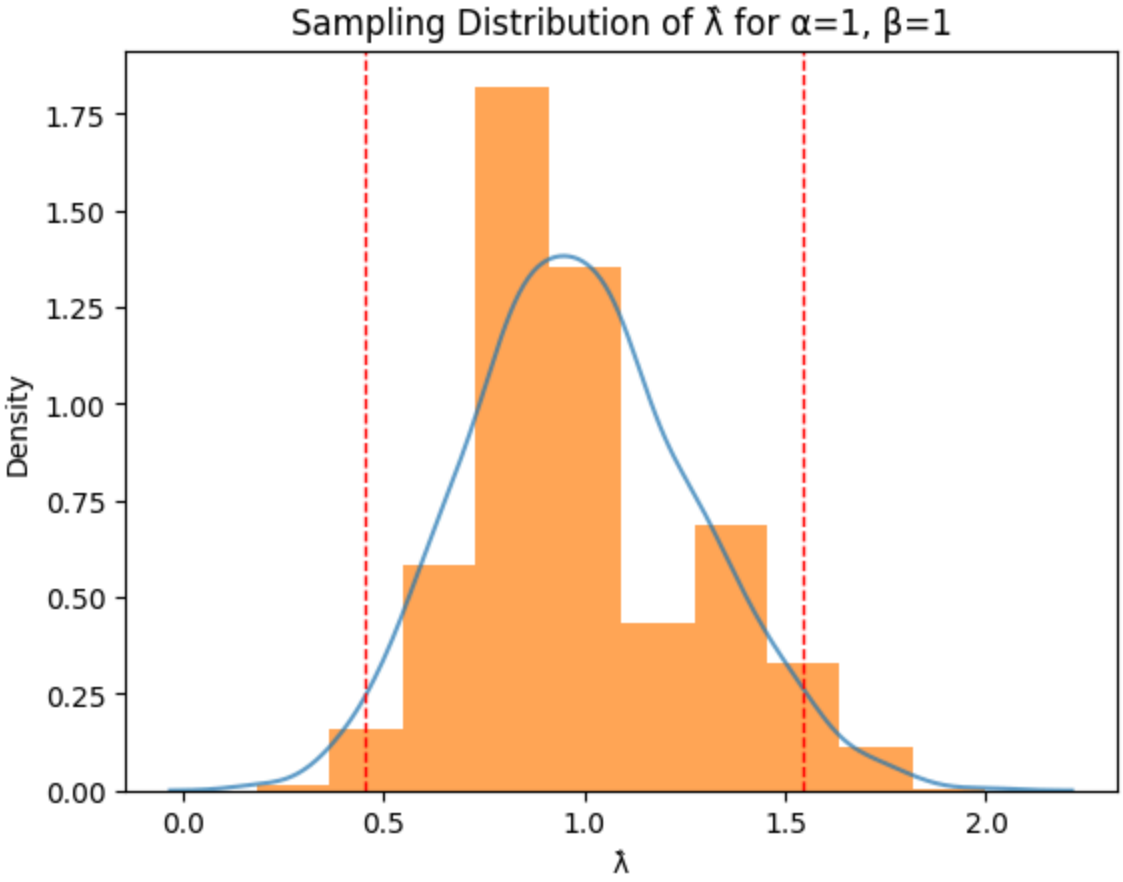
In Problem 1, we concluded that the estimator is unbiased when $\alpha = \lambda\beta$. We can check this by identifying the values of α, β, λ at the minima in the graphs and find that they do agree.

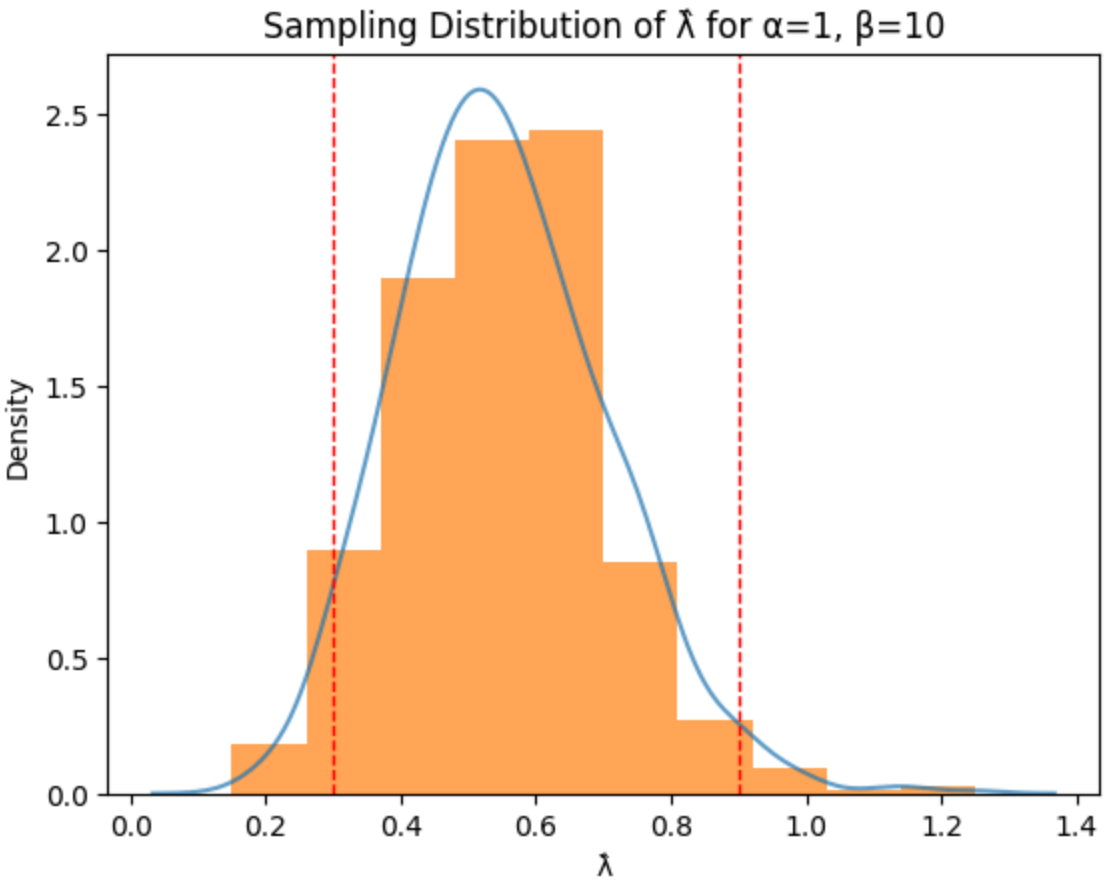
3

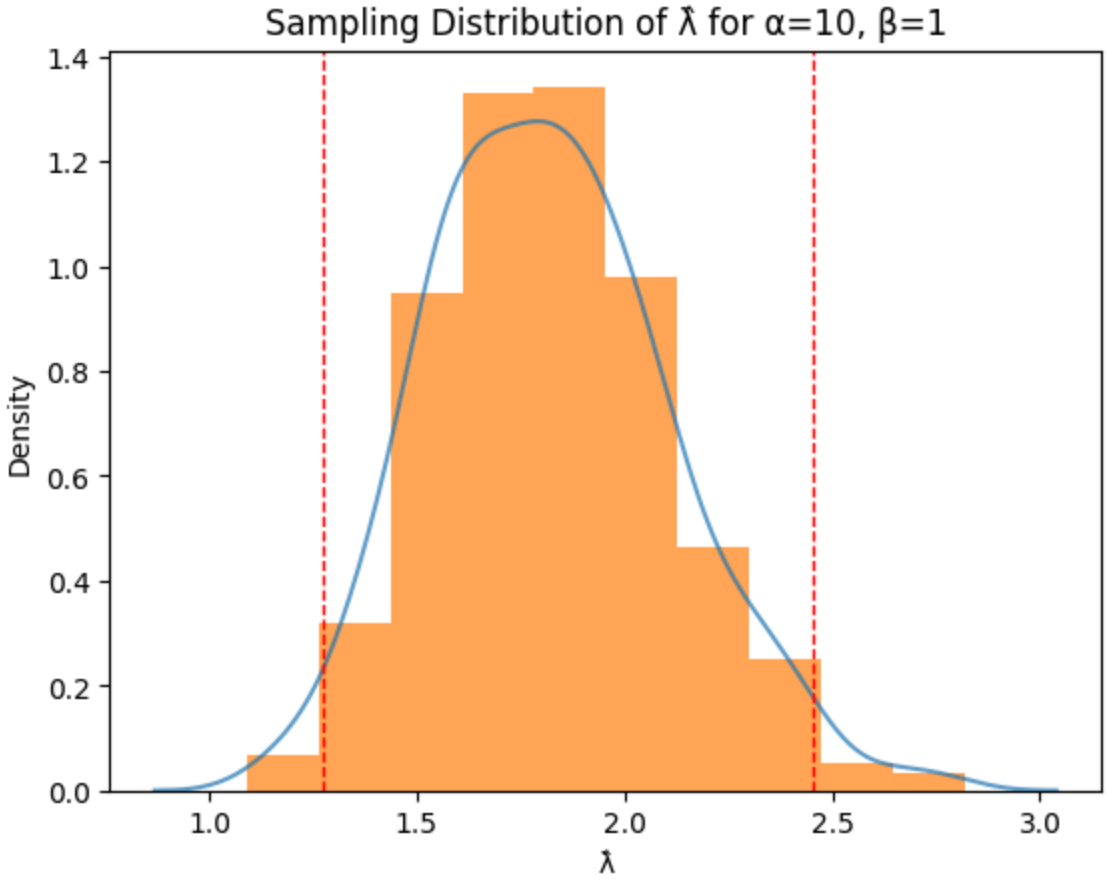
In [17]:
```
l = 1
N = 1000

ab_values = [(1, 1), (1, 10), (10, 1), (10, 10)]
n = 10

lambdahat_values = []

np.random.seed(0)
for a, b in ab_values:
    temp = []
    for _ in range(N):
        samples = np.random.poisson(l, n)
        lambdahat = (np.sum(samples) + a) / (n + b)
        temp.append(lambdahat)
    lambdahat_values.append(temp)
```

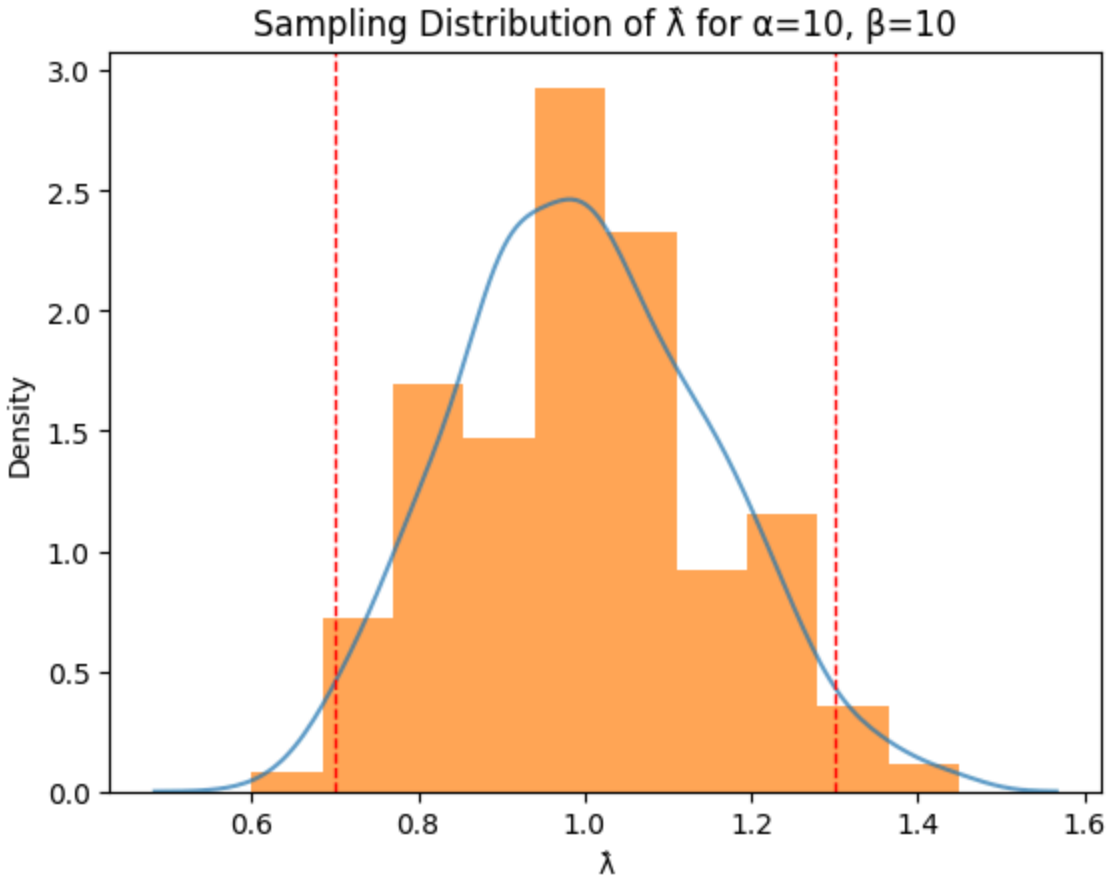
i

In [ ]:
```
import seaborn as sns

for (a, b), lambdahat in zip(ab_values, lambdahat_values):
    plt.figure()
    sns.kdeplot(lambdahat, alpha=0.7)
    plt.hist(lambdahat, bins=10, density=True, alpha=0.7)
    lower_bound = np.percentile(lambdahat, 2.5)
    upper_bound = np.percentile(lambdahat, 97.5)
    plt.axvline(lower_bound, color='red', linestyle='dashed', linewidth=1)
    plt.axvline(upper_bound, color='red', linestyle='dashed', linewidth=1)
    plt.title(f'Sampling Distribution of λˆfor α={a}, β={b}')
    plt.xlabel('λˆ')
    plt.ylabel('Density')
    plt.show()
```


Sampling Distribution of λ̂ for α=1, β=1
Density
λ̂
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
0.0
0.5
1.0
1.5
2.0


Sampling Distribution of λ̂ for α=1, β=10
Density
λ̂
0.0
0.5
1.0
1.5
2.0
2.5
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4


Sampling Distribution of λ̂ for α=10, β=1
Density
λ̂
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
1.0
1.5
2.0
2.5
3.0


Sampling Distribution of λ̂ for α=10, β=10
Density
λ̂
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.6
0.8
1.0
1.2
1.4
1.6

ii

In [19]:
```
for (a, b), lambdahat in zip(ab_values, lambdahat_values):
    empirical_mean = np.mean(lambdahat)
    empirical_variance = np.var(lambdahat)
    lower_bound = empirical_mean - 1.96 * np.sqrt(empirical_variance)
    upper_bound = empirical_mean + 1.96 * np.sqrt(empirical_variance)

    print(f'For α={a}, β={b}:')
    print(f'Empirical Mean: {empirical_mean}')
    print(f'Empirical Variance: {empirical_variance}')
    print(f'95% Wald CI: ({lower_bound}, {upper_bound})\n')
```

```
For α=1, β=1:
Empirical Mean: 0.9913636363636362
Empirical Variance: 0.08094194214876031
95% Wald CI: (0.43373780755127067, 1.548989465176002)

For α=1, β=10:
Empirical Mean: 0.5513499999999999
Empirical Variance: 0.0241206775
95% Wald CI: (0.24694567236322007, 0.8557543276367797)

For α=10, β=1:
Empirical Mean: 1.8191818181818182
Empirical Variance: 0.08648660330578511
95% Wald CI: (1.2427731651764145, 2.395590471187222)

For α=10, β=10:
Empirical Mean: 0.9993000000000001
Empirical Variance: 0.023769509999999997
95% Wald CI: (0.6971196736781166, 1.3014803263218835)
```

iii

When $\alpha = \beta$, the empirical mean is closest to the expected mean of 1, as then $\alpha = \lambda\beta$ because $\lambda = 1$. However, following our estimator for λ, all of the distributions agree to the expected values from problem 1.

iv

In [20]:
```
def expected_mean(a, b, n, l):
    return (n * l + a) / (n + b)

def expected_variance(a, b, n, l):
    return (n * l) / (n + b) ** 2

for (a, b) in ab_values:
```

```
    exp_mean = expected_mean(a, b, n, l)
    exp_variance = expected_variance(a, b, n, l)
    lower_bound = exp_mean - 1.96 * np.sqrt(exp_variance)
    upper_bound = exp_mean + 1.96 * np.sqrt(exp_variance)

    print(f'For α={a}, β={b}:')
    print(f'Expected Mean: {exp_mean}')
    print(f'Expected Variance: {exp_variance}')
    print(f'95% Wald CI: ({lower_bound}, {upper_bound})\n')
```

```
For α=1, β=1:
Expected Mean: 1.0
Expected Variance: 0.08264462809917356
95% Wald CI: (0.4365396169154524, 1.5634603830845477)

For α=1, β=10:
Expected Mean: 0.55
Expected Variance: 0.025
95% Wald CI: (0.24009678930349887, 0.8599032106965012)

For α=10, β=1:
Expected Mean: 1.8181818181818181
Expected Variance: 0.08264462809917356
95% Wald CI: (1.2547214350972706, 2.3816422012663656)

For α=10, β=10:
Expected Mean: 1.0
Expected Variance: 0.025
95% Wald CI: (0.6900967893034988, 1.3099032106965012)
```

These values from the theoretical conclusions in problem 1 agree with the values obtained from the Monte Carlo solutions.

v

There aren't exactly any peculiar features in the plots. It might be of note that the distributions are slightly right skewed, but this is expected as the Poisson distribution counts from 0 to infinity, and thus should be right skewed.