Question 1

$\alpha_k = \frac{1}{k}$

$Q^1[s_{17}, right] \leftarrow Q^0[s_{17}, right] + \alpha_k((r + 0.9\ max_{a'} Q^0[s_{18}, a']) - Q^0[s_{17}, right])$

$Q^1[s_{17}, right] \leftarrow 0 + \frac{1}{1} \times (2 + 0.9 \times 0 - 0) = 2$

$Q^1[s_{18}, up] \leftarrow Q^0[s_{18}, up] + \alpha_k((r + 0.9\ max_{a'} Q^0[s_{14}, a']) - Q^0[s_{18}, up])$

$Q^1[s_{18}, up] \leftarrow 0 + \frac{1}{1} \times (8 + 0.9 \times 0 - 0) = 8$

$Q^1[s_{14}, right] \leftarrow Q^0[s_{14}, right] + \alpha_k((r + 0.9\ max_{a'} Q^0[s_{15}, a']) - Q^0[s_{14}, right])$

$Q^1[s_{14}, right] \leftarrow 0 + \frac{1}{1} \times ((-6) + 0.9 \times 0 - 0) = -6$

(b)

$Q^1[s_{23}, up] \leftarrow Q^0[s_{23}, up] + \alpha_k((r + 0.9\ max_{a'} Q^1[s_{18}, a']) - Q^0[s_{23}, up])$

$Q^1[s_{23}, up] \leftarrow 0 + \frac{1}{1} \times (0 + 0.9 \times 8 - 0) = 7.2$

$Q^2[s_{18}, up] \leftarrow Q^1[s_{18}, up] + \alpha_k((r + 0.9\ max_{a'} Q^1[s_{14}, a']) - Q^1[s_{18}, up])$

$Q^2[s_{18}, up] \leftarrow 8 + \frac{1}{2} \times (0 + 0.9 \times 0 - 8) = 4$

$Q^2[s_{14}, right] \leftarrow Q^1[s_{14}, right] + \alpha_k((r + 0.9\ max_{a'} Q^1[s_{15}, a']) - Q^1[s_{14}, right])$

$Q^2[s_{14}, right] \leftarrow -6 + \frac{1}{2} \times (10 + 0.9 \times 0 - (-6)) = 2$

(c)

When running SARSA, $Q[s, a]$ is updated using $Q[s', a']$ instead of $max_{a'} Q[s', a']$. Therefore, for the previous part, the difference is that when updating $Q^2[s_{18}, up]$, SARSA will use $Q^1[s_{14}, right]$= -6 rather than $max_{a'} Q^1[s_{14}, a']$ = 0.

Question 2

Part a

(1)

In [1]:
```
import pandas as pd
import numpy as np
import math
from matplotlib import pyplot as plt
```

In [2]:
```
rs_samples = pd.read_csv('samples/rs_1.csv', names= ['data'])
rs_samples.head()
```

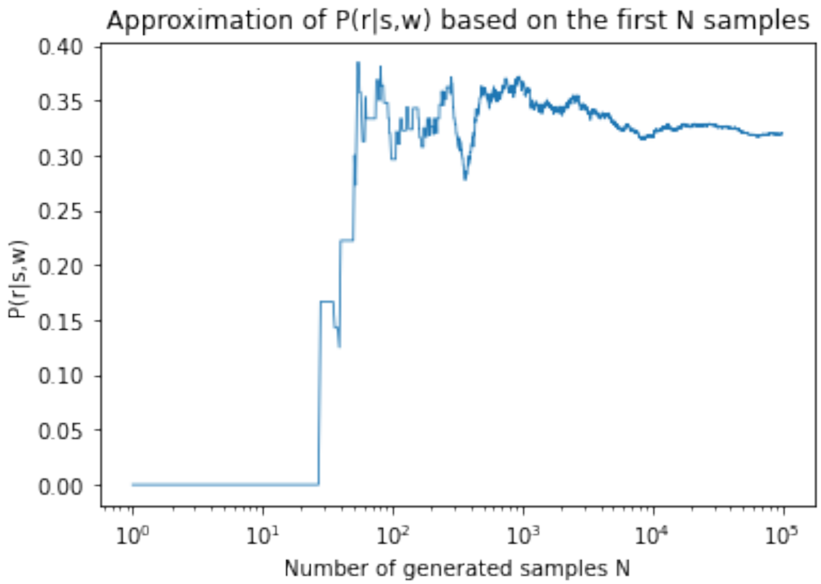
Out[2]:

	data
0	-1
1	2
2	-1
3	-1
4	-1

In [3]:
```
length = len(rs_samples)
```

In [4]:
```
def approxProbability(samples):
    acceptedNum = 0
    trueRainNum = 0
    y = [None]
    answer = []
    for i in range(1,length):
        #1: accepted sample with R = T
        if samples[i] == 1:
            acceptedNum += 1
            trueRainNum += 1
        #2: accepted sample with R = F
        elif samples[i] == 2:
            acceptedNum += 1
        y.append(trueRainNum / acceptedNum)

    # answer[0] = y
    answer.append(y)
    # answer[1] = acceptedNum
    answer.append(acceptedNum)
    return answer
```

In [5]:
```
x = rs_samples.index
y = approxProbability(list(rs_samples['data']))[0]
plt.plot(x,y,linewidth = 0.8)
plt.title('Approximation of P(r|s,w) based on the first N samples')
plt.xlabel('Number of generated samples N')
plt.ylabel('P(r|s,w)')
plt.xscale('log')
```



(2)

In [6]:
```
approx = y[length - 1]
print("The algorithm's approximation of P(r|s,w) using 100000 samples is: {:.6}".format(approx))
```

```
The algorithm's approximation of P(r|s,w) using 100000 samples is: 0.319814
```

Part b

(1)

Using Hoeffding's inequality to derive the tighest bound ϵ, we have:

$$
\begin{aligned}
2e^{-2n\epsilon^2} &< \delta \\
e^{-2n\epsilon^2} &< \frac{\delta}{2} \\
\frac{1}{e^{2n\epsilon^2}} &< \frac{\delta}{2} \\
\frac{2}{\delta} &< e^{2n\epsilon^2} \\
\ln\left(\frac{2}{\delta}\right) &< \ln\left(e^{2n\epsilon^2}\right) \\
\ln\left(\frac{2}{\delta}\right) &< 2n\epsilon^2 \\
\epsilon^2 &> \frac{\ln\left(\frac{2}{\delta}\right)}{2n} \\
\epsilon &> \sqrt{\frac{\ln\left(\frac{2}{\delta}\right)}{2n}}
\end{aligned}
$$

(2)

In [7]:

```
acceptedSampleNum = approxProbability(list(rs_samples['data']))[1]
print("At N=100000, there are n={} accepted samples.".format(acceptedSampleNum))
```

```
At N=100000, there are n=27910 accepted samples.
```

(3)

In [8]:

```
def computeEpsilon(n):
    delta = 0.05
    return math.sqrt(math.log(2/delta)/(2*n))
```

Given $n = 27910$ and $\delta = 0.05$, we have:

In [9]:
```
epsilon = computeEpsilon(27910)
print("The value of ε for n=27910 is {:.6}.".format(epsilon))
```

```
The value of ε for n=27910 is 0.00812928.
```

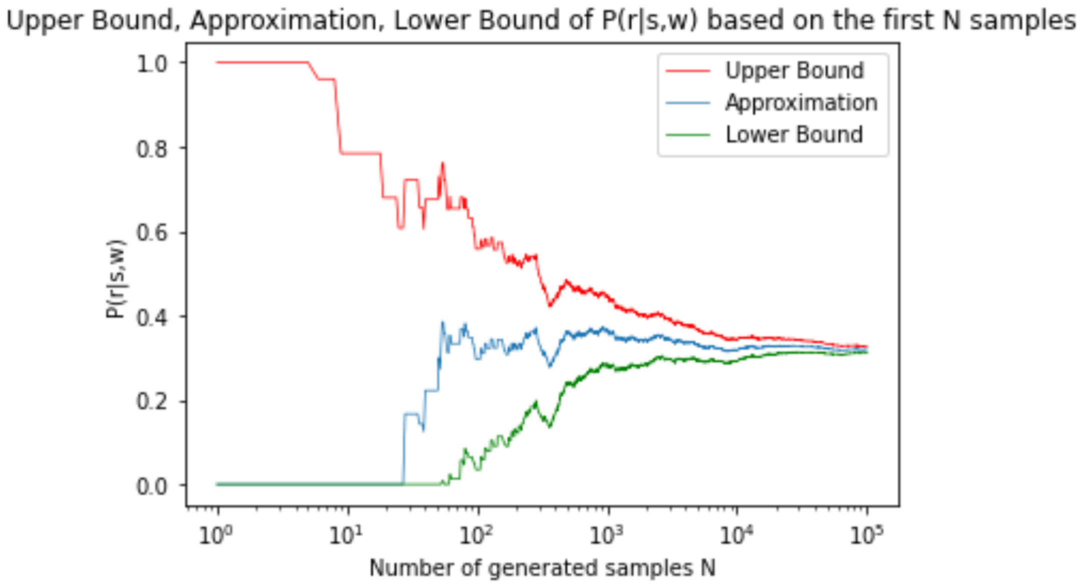
(4)

In [10]:
```
def upToPointEpsilons(samples):
    acceptedNum = 0
    e = [None]
    for i in range(1,length):
        if samples[i] == 1 or samples[i] == 2:
            acceptedNum += 1
        currEpsilon = computeEpsilon(acceptedNum,)
        e.append(currEpsilon)

    return e
```

In [11]:
```
def confidenceBounds(s, e):
    size = len(s)
    upperBound = [None]
    lowerBound = [None]
    bounds = []
    # probability should be in [0,1]
    for i in range(1,size):
        upperBound.append(min(s[i]+e[i],1))  # upper bound <=1
        lowerBound.append(max(s[i]-e[i],0))  # lower bound >=0

    # bounds[0] = upperBound
    bounds.append(upperBound)
    # bounds[1] = lowerBound
    bounds.append(lowerBound)

    return bounds
```

In [12]:
```
s = approxProbability(list(rs_samples['data']))[0]
e = upToPointEpsilons(list(rs_samples['data']))
upperBound = confidenceBounds(s,e)[0]
lowerBound = confidenceBounds(s,e)[1]
```

In [13]:
```
x = rs_samples.index

plt.plot(x,upperBound,linewidth = 0.7, c = 'r')
plt.plot(x,s,linewidth = 0.7)
plt.plot(x,lowerBound,linewidth = 0.7, c = 'g')

plt.legend(labels = ['Upper Bound', 'Approximation', 'Lower Bound'])
plt.title('Upper Bound, Approximation, Lower Bound of P(r|s,w) based on the first N samples')
plt.xlabel('Number of generated samples N')
plt.ylabel('P(r|s,w)')
plt.xscale('log')
```



Part c

(1)

In [14]:
```
lw_samples = pd.read_csv('samples/lw_1.csv', names= ['samples', 'weights'])
# 1 based index
lw_samples.index += 1
lw_samples.head()
```

Out[14]:

	samples	weights
1	2	0.450
2	1	0.495
3	2	0.090
4	2	0.450
5	2	0.450

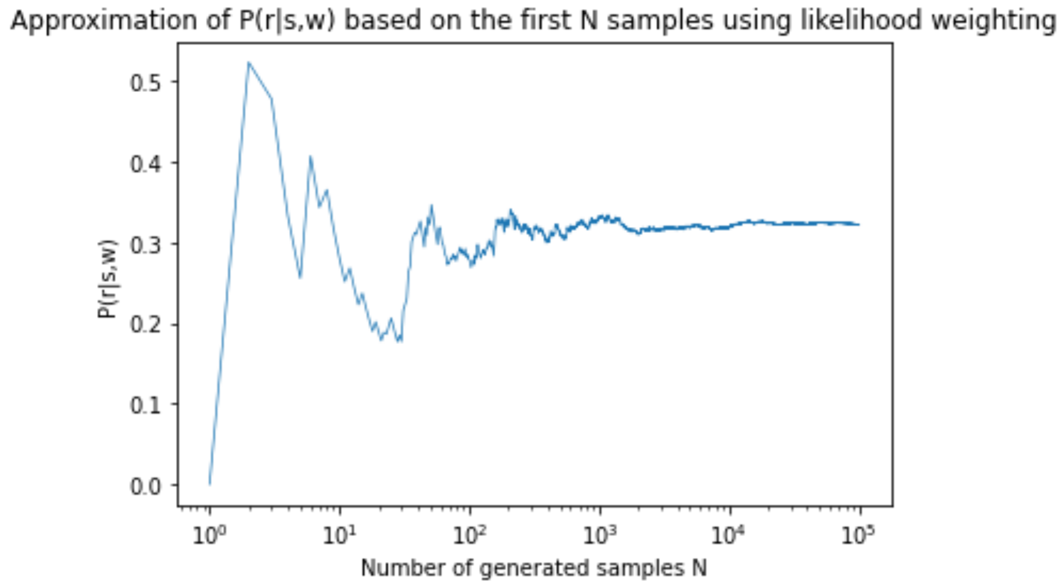
In [15]:
```
# 1 denotes R=T
# 2 denotes R=F
def lwProbability(samples, weights):
    tSampleWeights = 0
    totalWeights = 0
    y = []
    for i in range(0, len(samples)):
        totalWeights += weights[i]
        if samples[i] == 1:
            tSampleWeights += weights[i]
        y.append(tSampleWeights/totalWeights)
    return y
```

In [16]:
```
lw_y = lwProbability(list(lw_samples['samples']), list(lw_samples['weights']))
lwApprox = lw_y[len(lw_y) - 1]
print("The algorithm's approximation of P(r|s,w) using 100000 samples is: {:.6}".format(lwApprox))
```

```
The algorithm's approximation of P(r|s,w) using 100000 samples is: 0.321788
```

(2)

In [17]:
```
lw_x = lw_samples.index
lw_y = lwProbability(list(lw_samples['samples']), list(lw_samples['weights']))
plt.plot(lw_x,lw_y,linewidth = 0.7)
plt.title('Approximation of P(r|s,w) based on the first N samples using likelihood weighting')
plt.xlabel('Number of generated samples N')
plt.ylabel('P(r|s,w)')
plt.xscale('log')
```



(3)

The likelihood weigting algorithm seems to converge faster, especially when $N > 10^3$, whereas rejection sampling algorithm tends to converge when $N > 10^4$.

Question 3

(a)

We have the formula for forward message from filtering:

$$P(X_t|e_{0:t}) = \alpha\ P(e_t|X_t)P(X_t|e_{0:t-1})$$

$$= \alpha\ P(e_t|X_t)\sum_{x_{t-1}} P(X_t|x_{t-1})P(x_{t-1}|e_{0:t-1})$$

Hence, when t = 1, we have:

$$P(X_1|e_{0:0}) = P(X_1) = \sum_{x_0} P(X_1|x_0)P(x_0)$$

$$= <0.7, 0.3> \times 0.5 + <0.4, 0.6> \times 0.5$$

$$= <0.35, 0.15> + <0.2, 0.3>$$

$$= <0.55, 0.45>$$

$$P(X_1|e_1) = \alpha\ P(e_1|X_1)P(X_1)$$

$$= \alpha\ <0.8, 0.3> \times <0.55, 0.45>$$

$$= \alpha\ <0.44, 0.135>$$

$$= <0.7652173913, 0.2347826087>$$

When t = 2, we have:

$$P(X_2|e_{0:1}) = P(X_2|e_1) = \sum_{x_1} P(X_2|x_1)P(x_1|e_1)$$

$$= <0.7, 0.3> \times 0.7652173913 + <0.4, 0.6> \times 0.2347826087$$

$$= <0.535652173, 0.229565217> + <0.093913043, 0.140869565>$$

$$= <0.629565216, 0.370434782>$$

$$P(X_2|e_1, e_2) = \alpha\ P(e_2|X_2)P(X_2|e_1)$$

$$= \alpha <0.2, 0.7> \times <0.629565216, 0.370434782>$$

$$= \alpha <0.125913043, 0.259304347>$$

$$= <0.326862302, 0.673137697>$$

When t = 3, we have

$$P(X_3|e_{0:2}) = P(X_3|e_1, e_2) = \sum_{x_2} P(X_3|x_2)P(x_2|e_1, e_2)$$

$$= <0.7, 0.3> \times 0.326862302 + <0.4, 0.6> \times 0.673137697$$

$$= <0.228803611, 0.09805869> + <0.269255078, 0.403882618>$$

$$= <0.498058689, 0.501941308>$$

$$P(X_3|e_1, e_2, e_3) = \alpha\, P(e_3|X_3)P(X_t|e_1, e_2)$$

$$= \alpha\ <0.8, 0.3> \times <0.498058689, 0.501941308>$$

$$= \alpha\ <0.398446951, 0.150582392>$$

$$= <0.725729792, 0.274270207>$$

For the backward message, we have the formula:

$$P(e_{k+1:t}|X_k) = \sum_{x_{k+1}} P(e_{k+1}|x_{k+1})P(e_{k+2:t}|x_{k+1})P(x_{k+1}|X_k)$$

Combining forward message and backward message, we have:

$$P(X_k|e_{0:t}) = \alpha\ P(X_k|e_{0:k})\ P(e_{k+1:t}|X_k)$$

Hence, when k = 2, we have:

$P(e_3|X_2) = \sum_{x_3} P(e_3|x_3)P(|x_3)P(x_3|X_2)$

= 0.8×1× <0.7, 0.4> + 0.3×1× <0.3, 0.6>

= <0.56, 0.32> + <0.09, 0.18>

= <0.65, 0.5>

$P(X_2|e_1,\ e_2,\ e_3)$ = α $P(X_2|e_1,\ e_2)P(e_3|X_2)$

= α <0.326862302, 0.673137697> × <0.65, 0.5>

= α <0.212460496, 0.336568848>

= <0.386974755, 0.613025244>

When k = 1, we have:

$P(e_2,\ e_3|X_1) = \sum_{x_2} P(e_2|x_2)P(e_3|x_2)P(x_2|X_1)$

= 0.2× 0.65× <0.7, 0.4> + 0.7× 0.5× <0.3, 0.6>

= <0.091, 0.052> + <0.105, 0.21>

= <0.196, 0.262>

$P(X_1|e_1,\ e_2,\ e_3)$= α $P(X_1|e_1)P(e_2,\ e_3|X_1)$

= α <0.7652173913, 0.2347826087> × <0.196, 0.262>

= α <0.149982608, 0.061513043>

= <0.709152208, 0.290847791>

$P(X_1 = t|e_1,\ e_2,\ e_3)$ = 0.709152208

$P(X_2 = t|e_1,\ e_2,\ e_3)$ = 0.386974755

$P(X_3 = t|e_1,\ e_2,\ e_3)$ = 0.725729792

Hence, P(X_i= t|**e**) for i=1,2,3 is [0.709152208, 0.386974755, 0.725729792].

(b)
We have:

$$max_{x_1, ..., x_t} P(x_1, \dots x_t, X_{t+1}, e_{1:t+1}) = P(e_{t+1}|x_{t+1}) max_{x_t}[(P(X_{t+1}|x_t)\ m_{1:t}]$$

where

$$m_{1:t} = max_{x_1, ..., x_{t-1}} P(x_1, \dots , x_{t-1}, X_t, e_{1:t})$$

Using the result of filtering, when t = 1, we have:

$m_{1:1} = P(X_1|e_1)$ = <0.7652173913, 0.2347826087>

When t=2, we have:

$m_{1:2} = P(e_2|X_2)$ <max [$P(x_2|x_1)$ × 0.7652173913, $P(x_2|\sim x_1)$ × 0.2347826087],
max [$P(\sim x_2|x_1)$ × 0.7652173913, $P(\sim x_2|\sim x_1)$ × 0.2347826087]>
= <0.2, 0.7> × <max [0.7 × 0.7652173913, 0.4 × 0.2347826087],
max [0.3 × 0.7652173913, 0.6 × 0.2347826087]>
= <0.2, 0.7> × <max(0.535652173, 0.093913043),
max(0.229565217, 0.140871652)>
= <0.2, 0.7> × <0.535652173, 0.229565217>
= <0.107130434, 0.160695651>

When t=3, we have:

$m_{1:3} = P(e_3|X_3)$ <max [$P(x_3|x_2)$ × 0.107130434, $P(x_3|\sim x_2)$ × 0.160695651],
max [$P(\sim x_3|x_2)$ × 0.107130434, $P(\sim x_3|\sim x_2)$ × 0.160695651]>
= <0.8, 0.3> × <max [0.7 × 0.107130434, 0.4 × 0.160695651],
max [0.3 × 0.107130434, 0.6× 0.160695651]>
= <0.8, 0.3> × <max(0.074991303, 0.06427826), max(0.03213913, 0.09641739)>
= <0.8, 0.3> × <0.074991303, 0.09641739>
= <0.059993042, 0.028925217>

Since $P(X_3 = t)$>$P(X_3 = f)$, it is most likely that $X_3 = t$.
$P(X_3 = t)$ traces back to $P(x_3|x_2)$, which shows that $X_3 = t$ is more likely from $X_2 = t$.
$P(X_2 = t)$ traces back to $P(x_2|x_1)$, which indicates that $X_2 = t$ is more likely from $X_1 = t$.
Hence, the most likely sequence of states of variables X_1, X_2, X_3 is (t,t,t).