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

In [2]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

In [3]:
```
reads = pd.read_csv("/content/drive/My Drive/cm121/reads.tsv", sep='\t')
print(reads)
```

```
    observations  probability_of_error  error_truth
0              A              0.125650        False
1              T              0.092379        False
2              A              0.196982        False
3              T              0.063769        False
4              T              0.163563         True
..           ...                   ...          ...
995            A              0.037062        False
996            A              0.027094        False
997            A              0.146039        False
998            T              0.170114         True
999            T              0.038950        False

[1000 rows x 3 columns]
```

In [4]:
```
reads = reads.drop(columns=['error_truth'])
reads["probability_of_truth"] = 1 - reads['probability_of_error']
reads
```

Out[4]:

	observations	probability_of_error	probability_of_truth
0	A	0.125650	0.874350
1	T	0.092379	0.907621
2	A	0.196982	0.803018
3	T	0.063769	0.936231
4	T	0.163563	0.836437
...	...	...	...
995	A	0.037062	0.962938
996	A	0.027094	0.972906
997	A	0.146039	0.853961
998	T	0.170114	0.829886
999	T	0.038950	0.961050

1000 rows × 3 columns

Let E = {E1, E2, ..., E1000}

Find P(AA | E), P(AT | E), P(TT | E)

P(E): P(E | AA) P(AA) + P(E | AT) P(AT) + P(E | TT) P(TT)

In [5]:
```
def print_probabilities_using_log(n):
  if n==1000:
    indices = range(len(reads["observations"]))
  else:
    indices = np.random.choice(np.arange(1000), n)

  AA = []
  for i in indices:
    if reads["observations"][i] == "A":
      AA.append(reads["probability_of_truth"][i])
    else:
      AA.append(reads["probability_of_error"][i])

  TT = []
  for i in indices:
    if reads["observations"][i] == "T":
      TT.append(reads["probability_of_truth"][i])
    else:
      TT.append(reads["probability_of_error"][i])

  AT = []
  for i in indices:
    if reads["observations"][i] == "A":
      AT.append(0.5 * reads["probability_of_truth"][i] + 0.5 * reads["probability_of_error"][i])
    else:
      AT.append(0.5 * reads["probability_of_truth"][i] + 0.5 * reads["probability_of_error"][i])

  log_AA = list(map(lambda x: np.log(x), AA))
  log_TT = list(map(lambda x: np.log(x), TT))
  log_AT = list(map(lambda x: np.log(x), AT))

  log_p_data_given_AA = np.sum(log_AA)
  log_p_data_given_TT = np.sum(log_TT)
  # log_p_data_given_AT = np.sum(log_AT)
  log_p_data_given_AT = np.log(2**-len(AA)) # since p_data_given_AT reduces to 2^-n

  p_AA = 0.95**2
  p_TT = 0.05**2
  p_AT = 1 - p_AA - p_TT

  log_p_AA = np.log(p_AA)
  log_p_TT = np.log(p_TT)
  log_p_AT = np.log(p_AT)

  p_data_and_AA = np.exp(log_p_data_given_AA + log_p_AA)
  p_data_and_TT = np.exp(log_p_data_given_TT + log_p_TT)
  p_data_and_AT = np.exp(log_p_data_given_AT + log_p_AT)

  # p_data is a sum, cannot push log in, so must get values of summands first then add
  p_data = p_data_and_AA + p_data_and_TT + p_data_and_AT

  # can take log of whole probability then exp
  log_p_AA_given_data = log_p_data_given_AA + log_p_AA - np.log(p_data)
  log_p_TT_given_data = log_p_data_given_TT + log_p_TT - np.log(p_data)
  log_p_AT_given_data = log_p_data_given_AT + log_p_AT - np.log(p_data)

  return np.exp(log_p_AA_given_data), np.exp(log_p_TT_given_data), np.exp(log_p_AT_given_data)
```

In [6]:
```python
def print_probabilities(n):
  if n==1000:
    indices = range(len(reads["observations"]))
  else:
    indices = np.random.choice(np.arange(1000), n)

  AA = []
  for i in indices:
    if reads["observations"][i] == "A":
      AA.append(reads["probability_of_truth"][i])
    else:
      AA.append(reads["probability_of_error"][i])

  TT = []
  for i in indices:
    if reads["observations"][i] == "T":
      TT.append(reads["probability_of_truth"][i])
    else:
      TT.append(reads["probability_of_error"][i])

  AT = []
  for i in indices:
    if reads["observations"][i] == "A":
      AT.append(0.5 * reads["probability_of_truth"][i] + 0.5 * reads["probability_of_error"][i])
    else:
      AT.append(0.5 * reads["probability_of_truth"][i] + 0.5 * reads["probability_of_error"][i])

  p_data_given_AA = np.prod(AA)
  p_data_given_TT = np.prod(TT)
  p_data_given_AT = np.prod(AT)

  p_AA = 0.95**2
  p_TT = 0.05**2
  p_AT = 1 - p_AA - p_TT

  p_data_and_AA = p_data_given_AA * p_AA
  p_data_and_TT = p_data_given_TT * p_TT
  p_data_and_AT = p_data_given_AT * p_AT

  p_data = p_data_and_AA + p_data_and_TT + p_data_and_AT

  return p_data_and_AA / p_data,  p_data_and_TT / p_data,  p_data_and_AT / p_data
```

In [7]:
```python
print("P(AA | data), P(TT | data), P(AT | data):", print_probabilities_using_log(1000))
print("P(AA | data), P(TT | data), P(AT | data):", print_probabilities(1000))
```

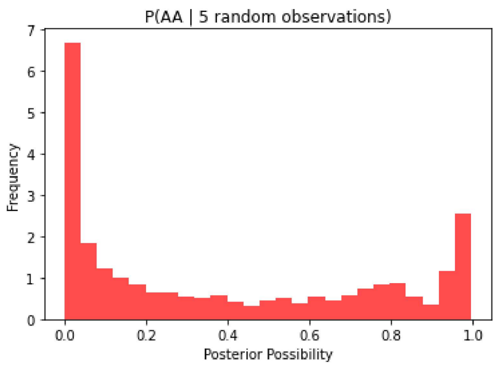
```
P(AA | data), P(TT | data), P(AT | data): (4.566267927369334e-300, 6.963414923204616e-290, 1.0)
P(AA | data), P(TT | data), P(AT | data): (0.0, 0.0, 1.0)
```

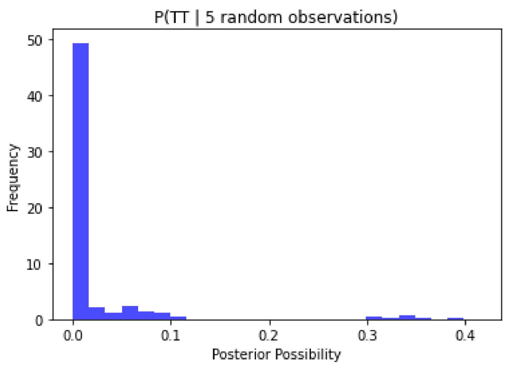
In [42]:
```python
## part c
print("P(AA | data), P(TT | data), P(AT | data):", print_probabilities_using_log(5))
```

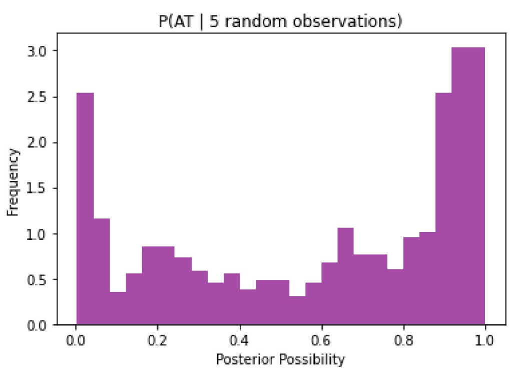
```
P(AA | data), P(TT | data), P(AT | data): (0.4422349524376797, 0.0002511003054347067, 0.5575139472568861)
```

In [49]:
```python
## part d
aa_hist = []
tt_hist = []
at_hist = []
for i in range(1000):
  aa, tt, at = print_probabilities_using_log(5)
  aa_hist.append(aa)
  tt_hist.append(tt)
  at_hist.append(at)
```

In [50]:
```
plt.hist(aa_hist, 25, density = 1, color ='red', alpha = 0.7)
plt.xlabel('Posterior Possibility')
plt.ylabel('Frequency')

plt.title('P(AA | 5 random observations)')
plt.show()

plt.hist(tt_hist, 25, density = 1, color ='blue', alpha = 0.7)
plt.xlabel('Posterior Possibility')
plt.ylabel('Frequency')

plt.title('P(TT | 5 random observations)')
plt.show()

plt.hist(at_hist, 25, density = 1, color ='purple', alpha = 0.7)
plt.xlabel('Posterior Possibility')
plt.ylabel('Frequency')

plt.title('P(AT | 5 random observations)')
plt.show()
```





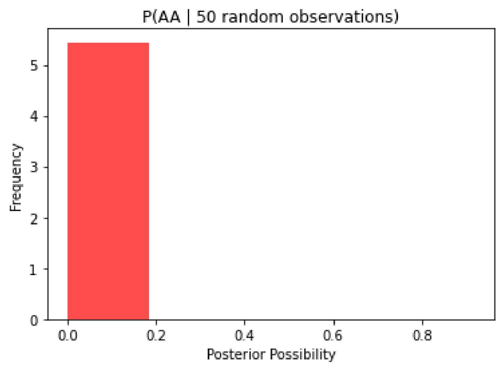


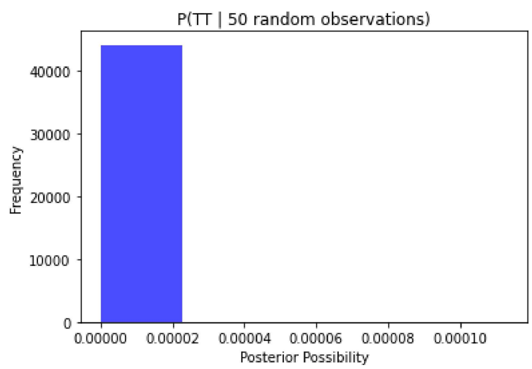
In [51]:
```
print("P(AA | 5 obs) mean:", np.mean(aa_hist))
print("P(AA | 5 obs) stddev:", np.std(aa_hist))
print("P(TT | 5 obs) mean:", np.mean(tt_hist))
print("P(TT | 5 obs) stddev:", np.std(tt_hist))
print("P(AT | 5 obs) mean:", np.mean(at_hist))
print("P(AT | 5 obs) stddev:", np.std(at_hist))
```

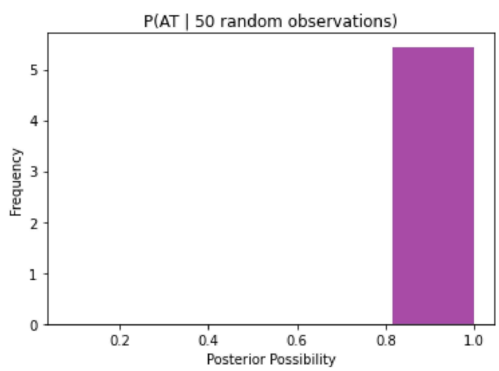
```
P(AA | 5 obs) mean: 0.38864735486345364
P(AA | 5 obs) stddev: 0.3658517620690793
P(TT | 5 obs) mean: 0.02160467603008359
P(TT | 5 obs) stddev: 0.06416173550246762
P(AT | 5 obs) mean: 0.5897479691064628
P(AT | 5 obs) stddev: 0.3491135684915129
```

In [52]:
```
## part e
aa_hist_50 = []
tt_hist_50 = []
at_hist_50 = []
for i in range(1000):
  aa, tt, at = print_probabilities_using_log(50)
  aa_hist_50.append(aa)
  tt_hist_50.append(tt)
  at_hist_50.append(at)
```

In [57]:
```
plt.hist(aa_hist_50, 5, density = 1, color ='red', alpha = 0.7)
plt.xlabel('Posterior Possibility')
plt.ylabel('Frequency')

plt.title('P(AA | 50 random observations)')
plt.show()

plt.hist(tt_hist_50, 5, density = 1, color ='blue', alpha = 0.7)
plt.xlabel('Posterior Possibility')
plt.ylabel('Frequency')

plt.title('P(TT | 50 random observations)')
plt.show()

plt.hist(at_hist_50, 5, density = 1, color ='purple', alpha = 0.7)
plt.xlabel('Posterior Possibility')
plt.ylabel('Frequency')

plt.title('P(AT | 50 random observations)')
plt.show()
```







In [58]:
```
print("P(AA | 50 obs) mean:", np.mean(aa_hist_50))
print("P(AA | 50 obs) stddev:", np.std(aa_hist_50))
print("P(TT | 50 obs) mean:", np.mean(tt_hist_50))
print("P(TT | 50 obs) stddev:", np.std(tt_hist_50))
print("P(AT | 50 obs) mean:", np.mean(at_hist_50))
print("P(AT | 50 obs) stddev:", np.std(at_hist_50))
```

```
P(AA | 50 obs) mean: 0.0009612494416033293
P(AA | 50 obs) stddev: 0.02902158884275641
P(TT | 50 obs) mean: 1.322590473860504e-07
P(TT | 50 obs) stddev: 3.5897922372511343e-06
P(AT | 50 obs) mean: 0.9990386182993493
P(AT | 50 obs) stddev: 0.02902158468410601
```

In [18]: