Assignment 01 (CCNS 698)

1.1 (a) Sample Space

for two coins being tossed simultaneously, each coin can land either heads (H) or tails (T).

so the sample space Ω

$$\Omega = \{ (H,H), (H,T), (T,H), (T,T) \}$$

(b) Event space is set of all possible events and events are subset of sample space hence event space is power set of sample space.

$$E = \{ \{\}, \{H,H\}, \{T,H\}, \{H,T\}, \{T,T\}, \{HH, HT\}, \{HH, TH\}, \{HH, TT\}, \{HT, TH\},$$
$$\{TH, TT\}, \{HT, TT\}, \{HH, HT, TH\}, \{HH, HT, TT\}, \{HH, TH, TT\},$$
$$\{HT, TH, TT\}, \{HH, HT, TH, TT\} \}$$

(c) (i) $P(\{(H,H)\}) = P(\{(H,T)\}) = P(\{(T,H)\}) = P(\{(T,T)\}) = \frac{1}{4}$

(ii) at least one head appears.

$\{(H,H), (H,T), (T,H)\} = 3$

$$P = \frac{\text{fav. outcomes}}{\text{Total outcomes}} = 3/4 = 0.75$$

(iii) Exactly one head appears.

$$P = \frac{(H,T), (T,H)}{(H,T), (T,H), (T,T), (H,H)} = 2/4 = 1/2 = 0.5$$

Part 2 Discrete random variables.

2.1 $f(n,n,p) = \frac{n!}{k!\,(n-k)!} p^k (1-p)^{(n-k)}$

K = number of correctly recognized words.

n = total no. of words.

P(rec. word) = 0.9

P(Not rec. word) = 0.

$$P(x=45) = \frac{50!}{45!\,(50-45)!} (0.9)^{45} (0.1)^5$$

$$= \frac{50!}{45!\,5!}(0.9)^{45}(0.1)^{5}$$

$$\simeq \cancel{2118760\,(0.9)^{45}(0.1)^{5}}$$

$$\simeq 0.\cancel{27}184 = \cancel{0.27}$$

2.2

Sol^n

$$f(k,\lambda) = \frac{\lambda^k e^{-\lambda}}{k!} \quad ; \quad \lambda = 10$$

(a) Zero accident $= f(0,10) = \dfrac{(10)^0 e^{-10}}{0!}$

$$= e^{-10}$$

$$\simeq 4.54 \times 10^{-5}$$

(b) we need to find the probability for $8 \le k \le 9$

$$P(8 \le x \le 9) = P(x=8) + P(x=9)$$

$$= \frac{10^8 e^{-10}}{8!} + \frac{10^9 e^{-10}}{9!}$$

$$= \frac{10^8 e^{-10}}{8!}\left(1 + \frac{10}{9}\right)$$

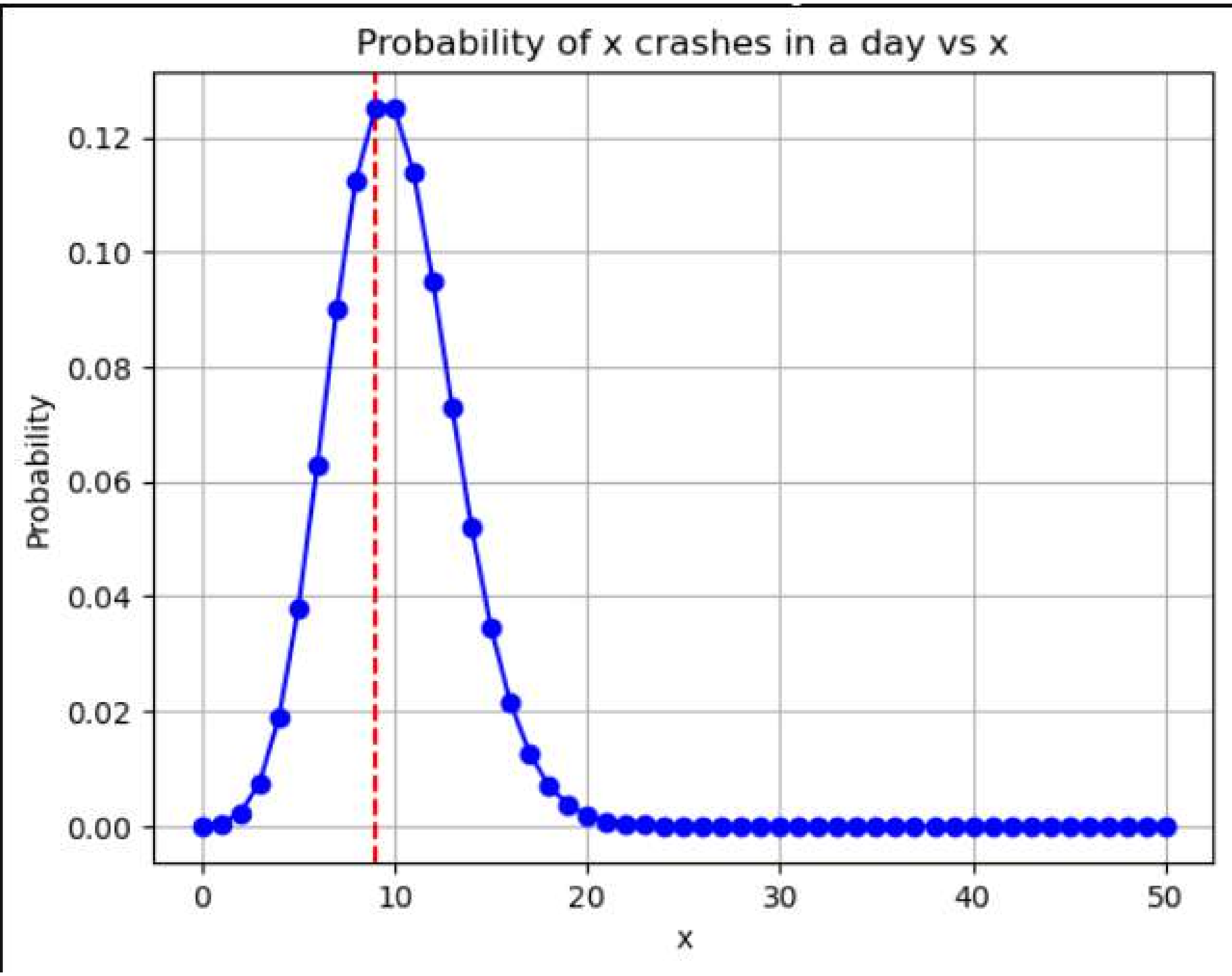
$$= \frac{19 \times 10^8 \times e^{-10}}{9!}$$

$$\simeq 0.2377 = 0.24$$

(c)

2.2(c) sol.
code and graph

```
import math
import matplotlib.pyplot as plt
import numpy as np
def p(x):
    return float(pow(10, x) * pow(np.e, -10)) /(math.factorial(x))
x = np.arange(0,51,1)
print(x)
vectorize = np.vectorize(p)
y = vectorize(x)
print(y)

max_x = x[np.argmax(y)]
max_y = np.max(y)

plt.axvline(x=max_x, color='r', linestyle='--', label=f'Max f(x) at x={max_x}')
plt.plot(x, y, marker='o', linestyle='-', color='b')
plt.xlabel('x')
plt.ylabel('Probability')
plt.title('Probability of x crashes in a day vs x')
plt.grid(True)
plt.show()
```


Probability of x crashes in a day vs x
Probability
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0
10
20
30
40
50
x

Part 3 $f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$

(a) probability density at $x=0$ Given $\mu=1$ & $\sigma=1$

$$f(0) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(0-1)^2}{2\cdot 1^2}}$$

$$= \frac{1}{\sqrt{2\pi}} e^{-0.5} \simeq 0.242$$

(b) $f(1) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(1-0)^2}{2(1)^2}}$ $\underline{(\mu=0, \sigma=1)}$

$$\simeq \frac{1}{\sqrt{2\pi}} e^{-0.5}$$

$$\simeq 0.242$$

(c) probability outcomes between x_2 & x_3

Given: $P(x_1 \leq X \leq x_2) = 0.3$
$P(x_1 \leq X \leq x_3) = 0.45$) Mutually Exlusive.

$$P(x_1 \leq X \leq x_3) = P(x_1 \leq X \leq x_2) + P(x_2 \leq X \leq x_3)$$

$$P(x_2 \leq X \leq x_3) = P(x_1 \leq X \leq x_3) - P(x_1 \leq X \leq x_2)$$

$$= 0.45 - 0.30$$

$$= 0.15$$

Part 4

$L(\theta|x) = f(x, \theta)$ x is fixed.

recognition time for 5 strings $= 303, 443, \overset{220}{\vee} 560, 880$ ms

$$f(x, \mu) = \frac{1}{x\sqrt{2\pi}} e^{-\frac{(\log x - \mu)^2}{2}}$$

$x = 220$

$$L(\theta/220) = f(220, \theta)$$
$$= \frac{1}{220\sqrt{2\pi}} e^{-\frac{(\log 220 - \mu)^2}{2}}$$

b) $x = [303.25, 443, 229\ 560, 880]$

$$L(\mu/x) = f(x, \mu) = \prod_{i=1}^{5} f(x[i], \mu)$$

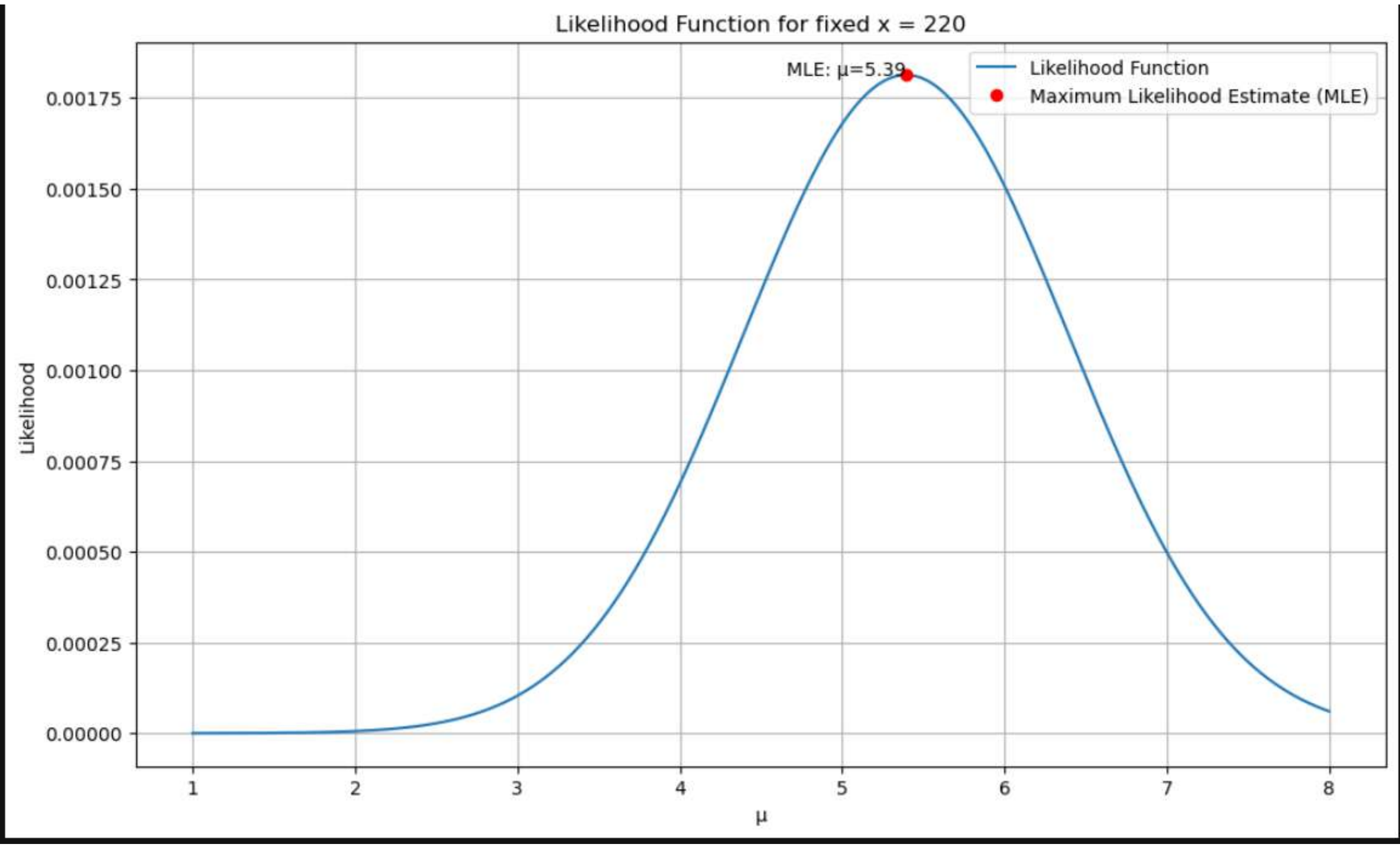
$$= \frac{1}{\sqrt{2\pi}\left(\prod_{i=1}^{n} x[i]\right)} e^{-\frac{\sum_{i=1}^{n}(\log x[i] - \mu)^2}{2}}$$

c) from the Graph

$L(\mu/x)$ max at $\mu \simeq 6$

4.1(a)
code and graph

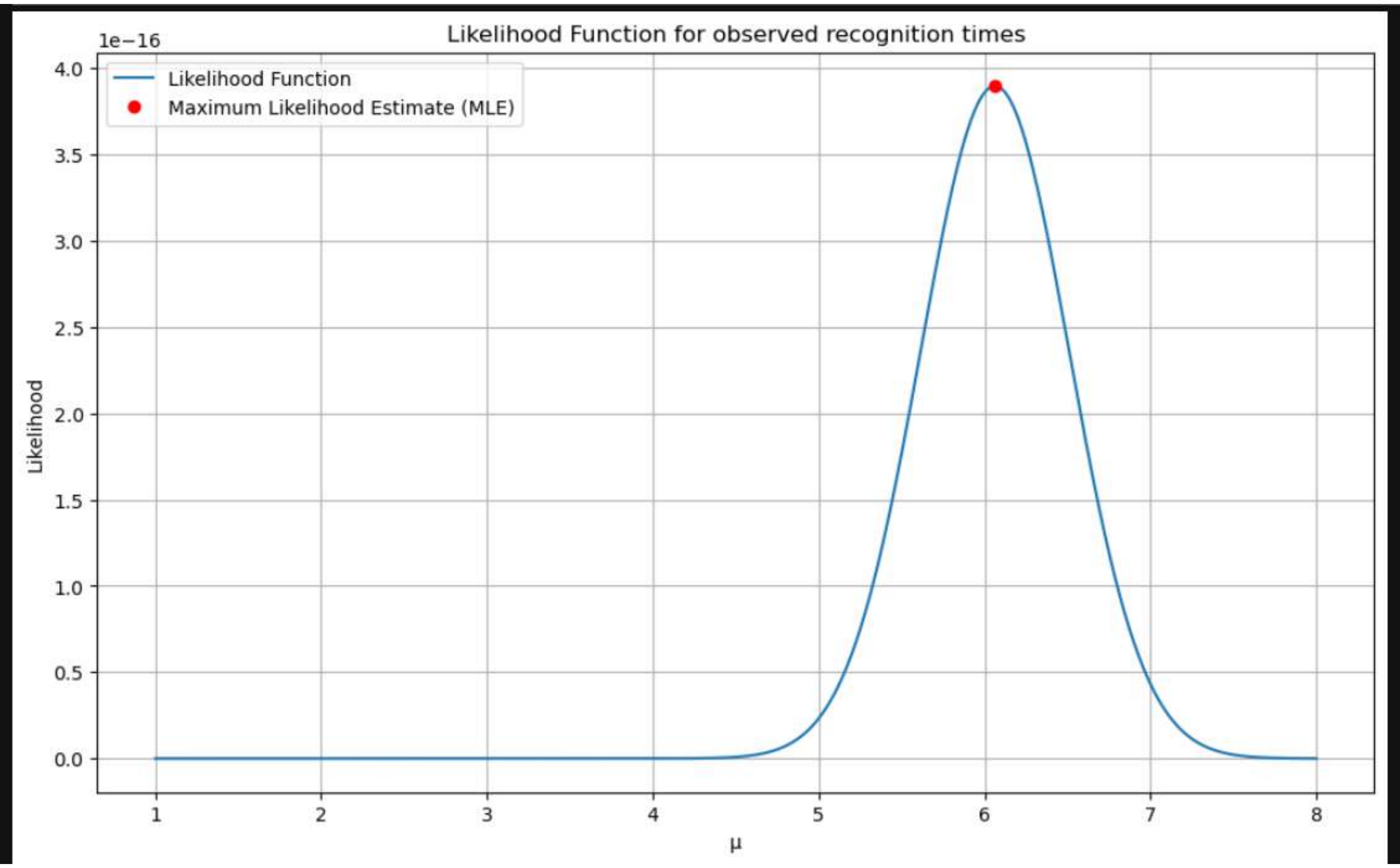
```
import numpy as np
import matplotlib.pyplot as plt

# Define the likelihood function f(x, μ) given x and μ
def likelihood(x, mu):
    return (1 / (x * np.sqrt(2 * np.pi))) * np.exp(-((np.log(x) - mu) ** 2) / 2)

# Given x = 220
x_fixed = 220

# Choose a suitable range of values for μ
mu_values = np.linspace(1, 8, 1000)  # Expanded the range for full graph

# Calculate the likelihood for each value of μ
likelihood_values = likelihood(x_fixed, mu_values)

# Find the index of the maximum likelihood value
max_index = np.argmax(likelihood_values)

# Extract the corresponding value of mu
mu_max_likelihood = mu_values[max_index]

# Plot the graph
plt.figure(figsize=(12, 7))
plt.plot(mu_values, likelihood_values, label='Likelihood Function')
plt.plot(mu_max_likelihood, likelihood_values[max_index], 'ro', label='Maximum Likelihood Estimate (MLE)')
plt.text(mu_max_likelihood, likelihood_values[max_index], f'MLE: μ={mu_max_likelihood:.2f}', ha='right')
plt.xlabel('μ')
plt.ylabel('Likelihood')
plt.title('Likelihood Function for fixed x = 220')
plt.legend()
plt.grid(True)
plt.show()
```



Likelihood Function for fixed x = 220
MLE: μ=5.39
Likelihood Function
Maximum Likelihood Estimate (MLE)
Likelihood
0.00000
0.00025
0.00050
0.00075
0.00100
0.00125
0.00150
0.00175
1
2
3
4
5
6
7
8
μ

4.1(b)code and graph

```python
import numpy as np
import matplotlib.pyplot as plt

# Define the likelihood function f(x, μ) given x and μ
def likelihood(x, mu):
    n = len(x)
    log_likelihood = -np.sum((np.log(x) - mu)**2)
    return (1 / (np.prod(x) * np.sqrt(2 * np.pi)**n)) * np.exp(log_likelihood / 2)

# Given observed recognition times
x_observed = [303.25, 443, 220, 560, 880]

# Choose a suitable range of values for μ
mu_values = np.linspace(1, 8, 1000)  # Adjust the range as needed

# Calculate the likelihood for each value of μ
likelihood_values = [likelihood(x_observed, mu) for mu in mu_values]

# Find the index of the maximum likelihood value
max_index = np.argmax(likelihood_values)

# Extract the corresponding value of mu
mu_max_likelihood = mu_values[max_index]

# Plot the graph
plt.figure(figsize=(12, 7))
plt.plot(mu_values, likelihood_values, label='Likelihood Function')
plt.plot(mu_max_likelihood, likelihood_values[max_index], 'ro', label='Maximum Likelihood Estimate (MLE)')
plt.xlabel('μ')
plt.ylabel('Likelihood')
plt.title('Likelihood Function for observed recognition times')
plt.legend()
plt.grid(True)
plt.show()
```


Likelihood Function for observed recognition times
1e−16
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Likelihood
Likelihood Function
Maximum Likelihood Estimate (MLE)
1
2
3
4
5
6
7
8
μ