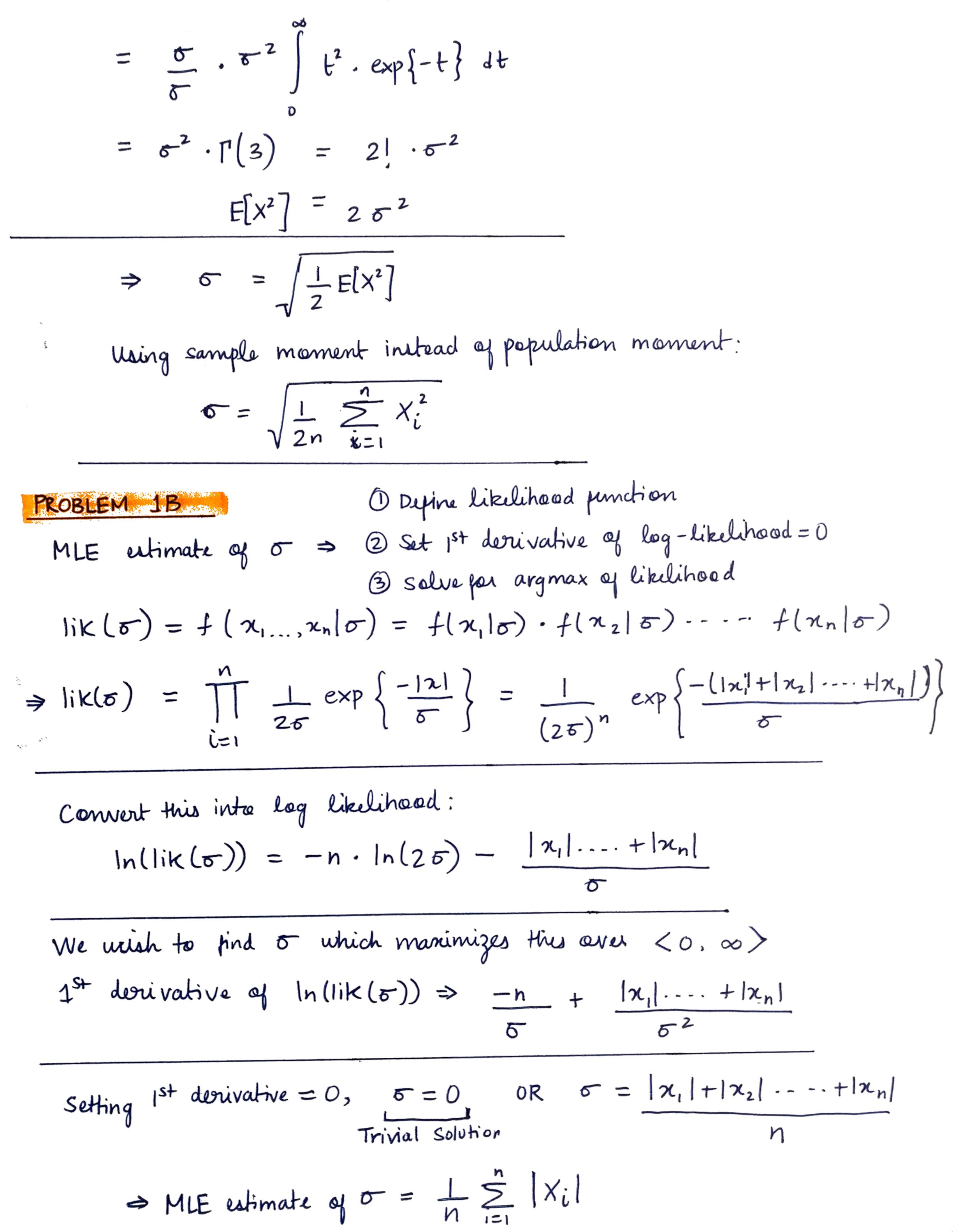
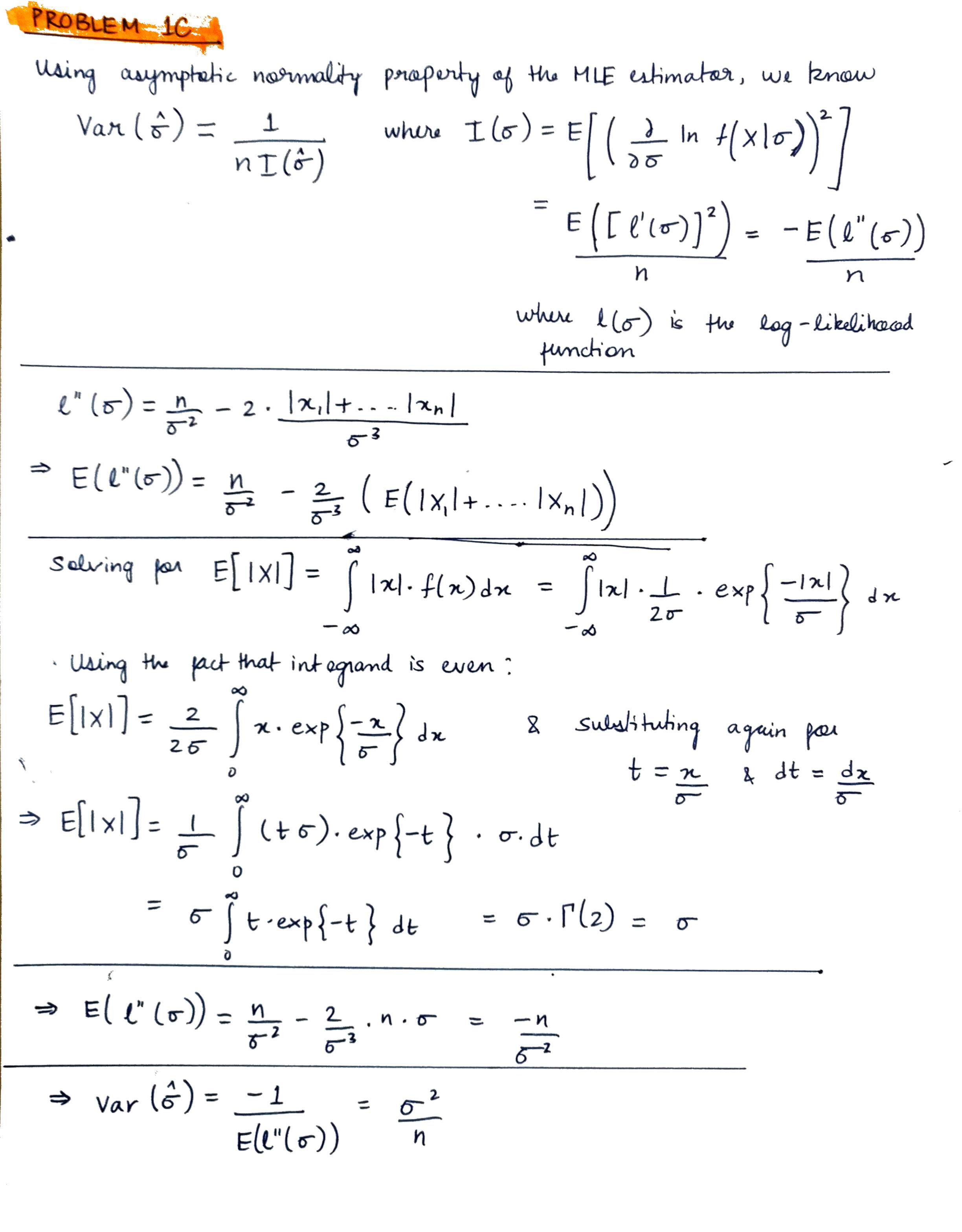
**Problem 1A & 1B**

**A piece of paper with writing

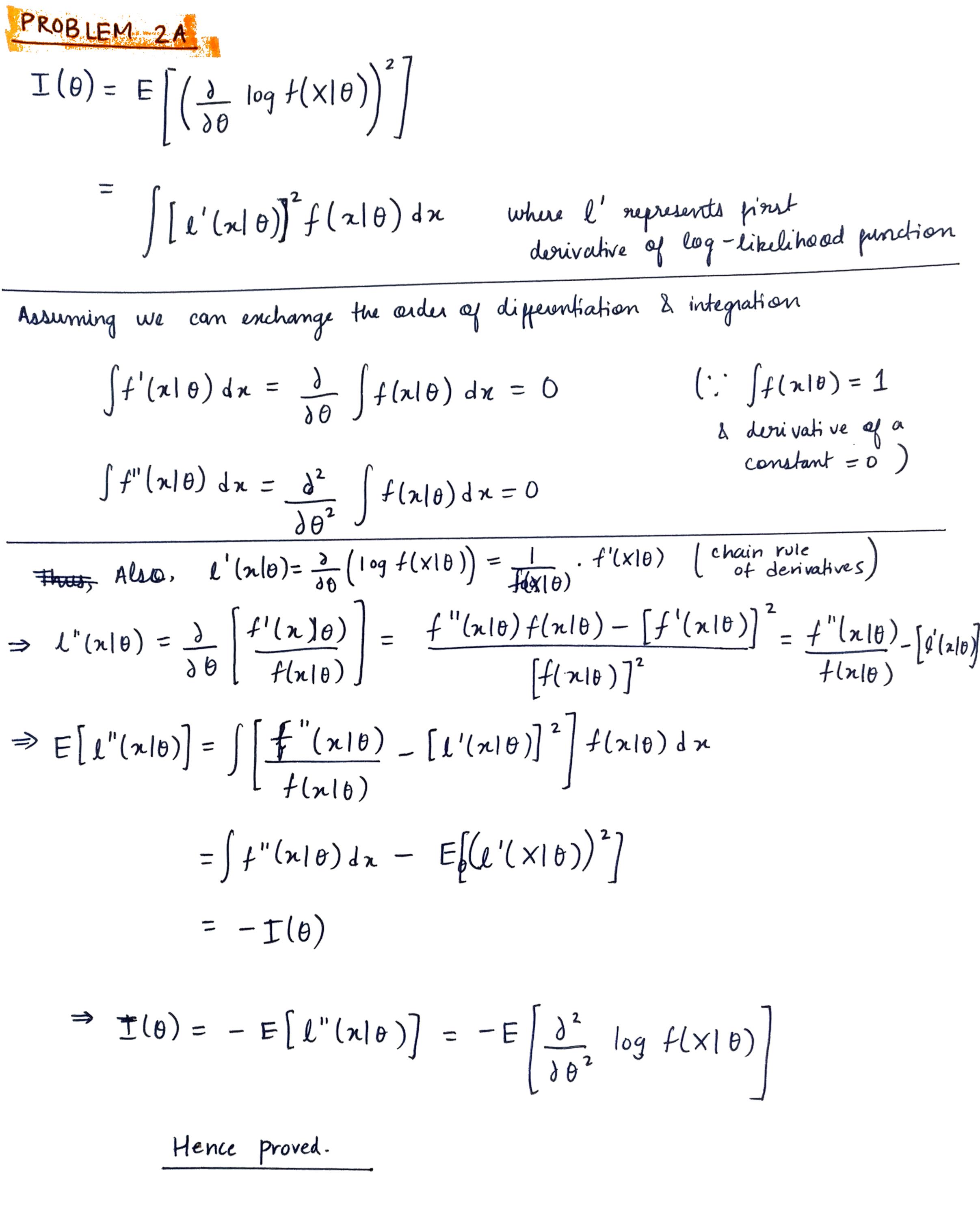
Description automatically generated with medium confidence**

****

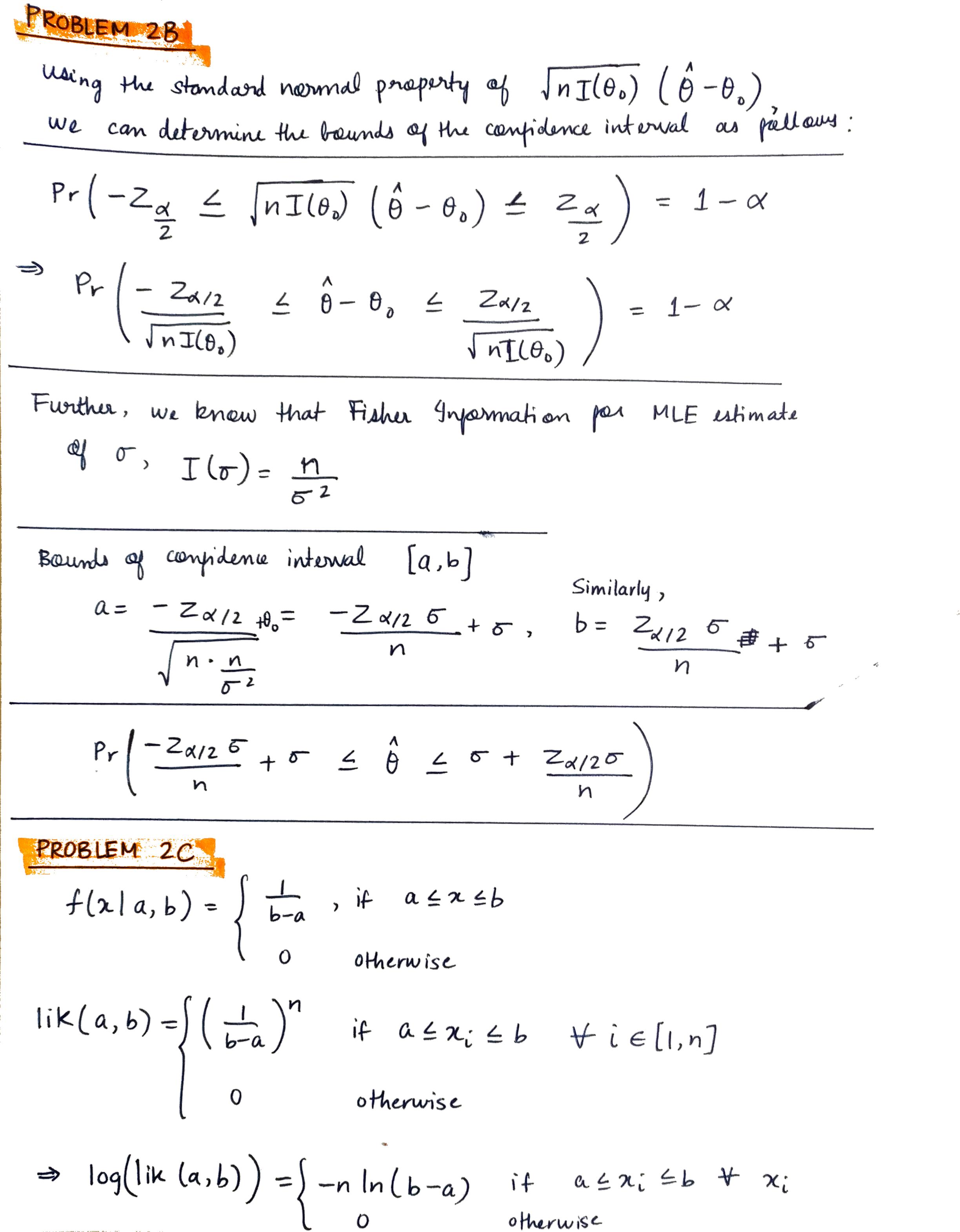
**Problem 1C**

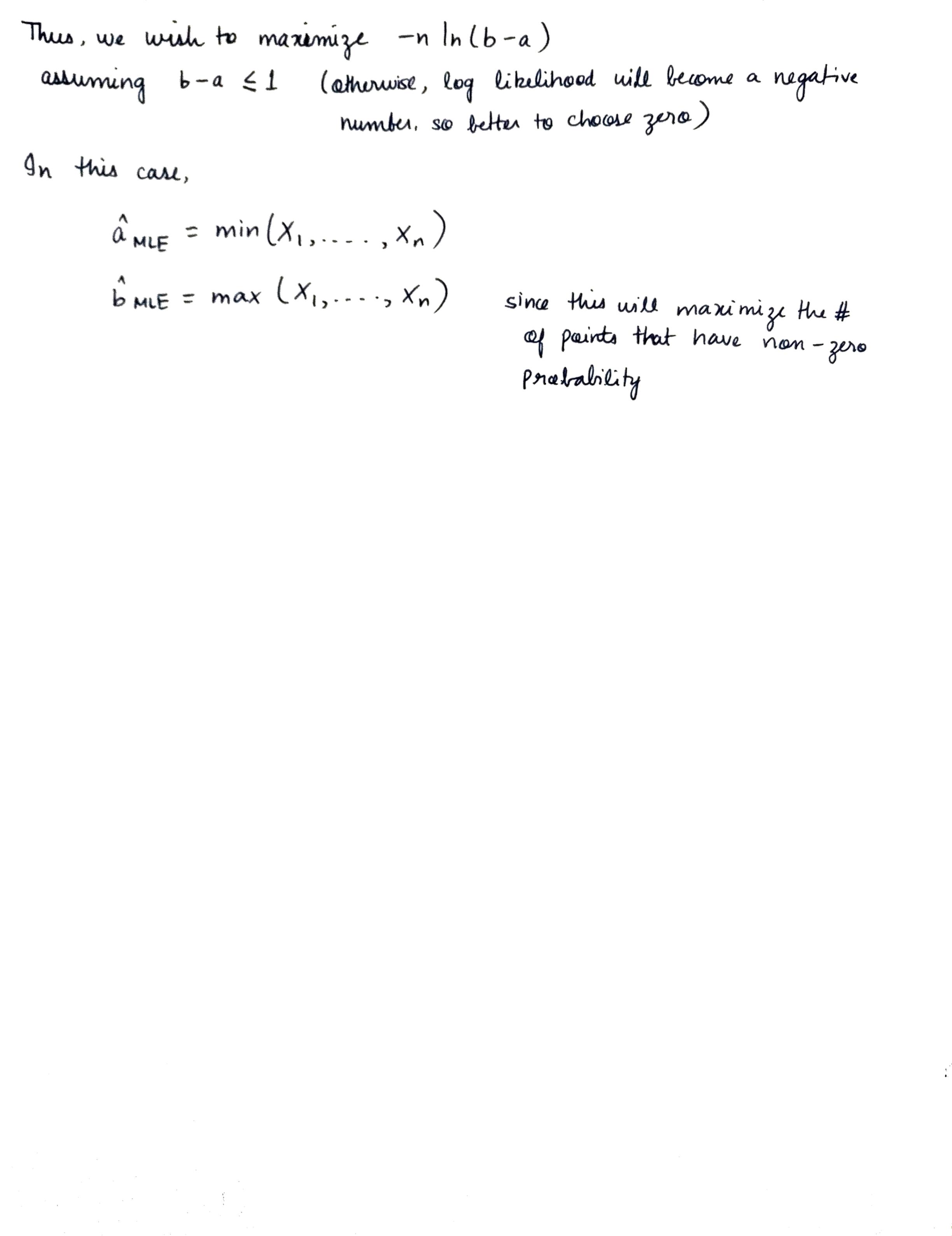
****

**Problem 2A**

****

**Problem 2B & 2C**

****

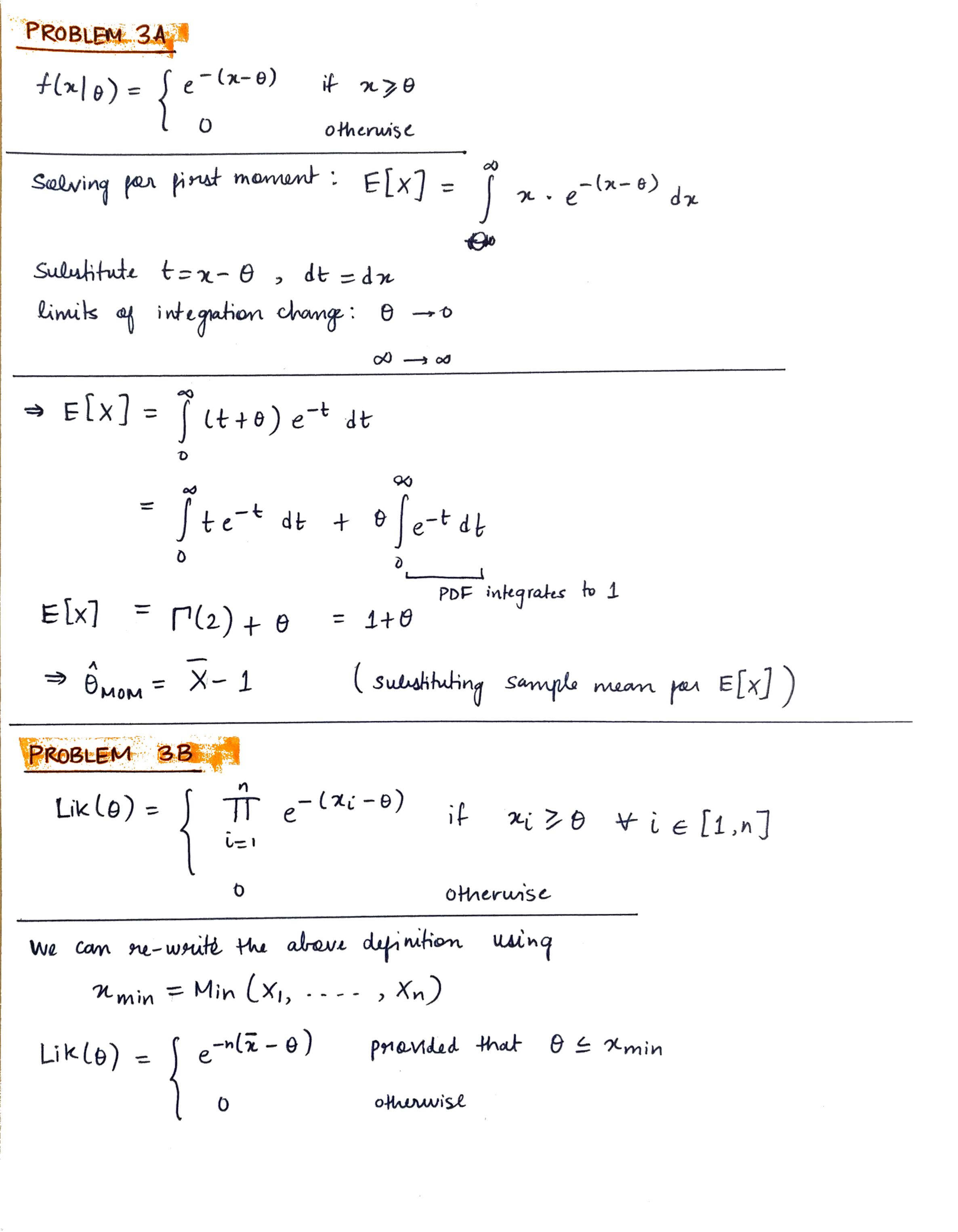
****

**Problem 3A**

**Text, letter

Description automatically generated**

**Problem 3B**

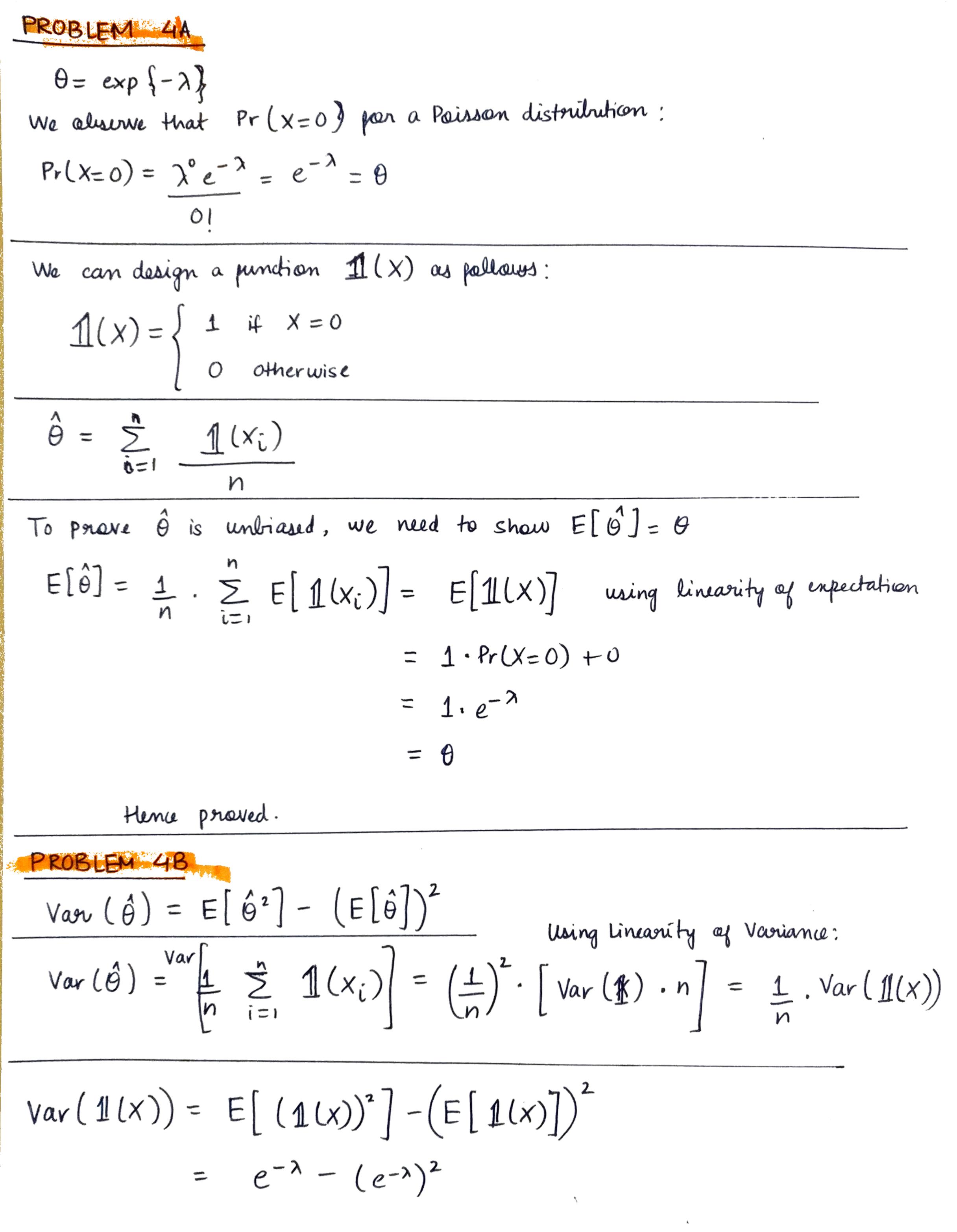
****

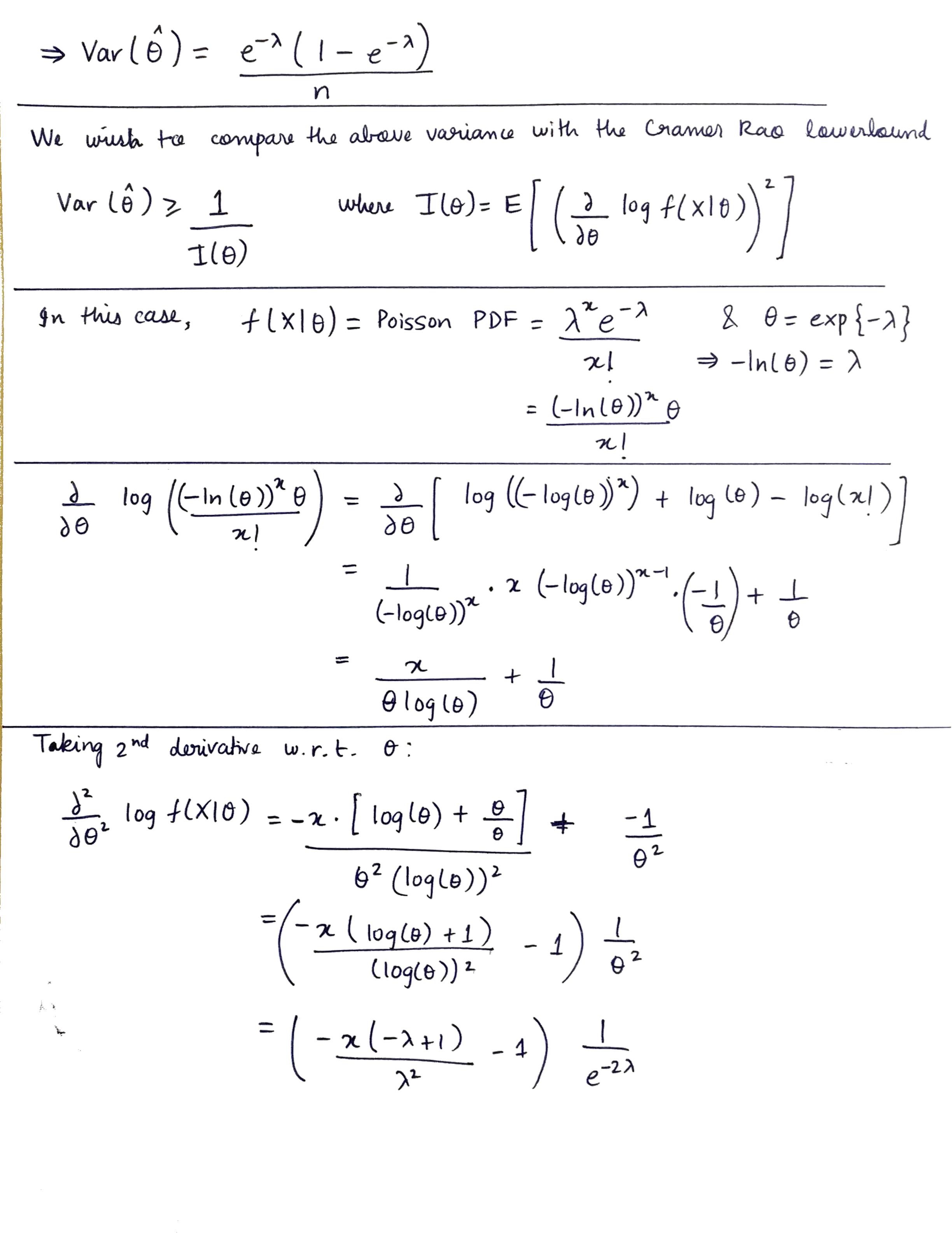
**Problem 4A**

**Text, letter

Description automatically generated**

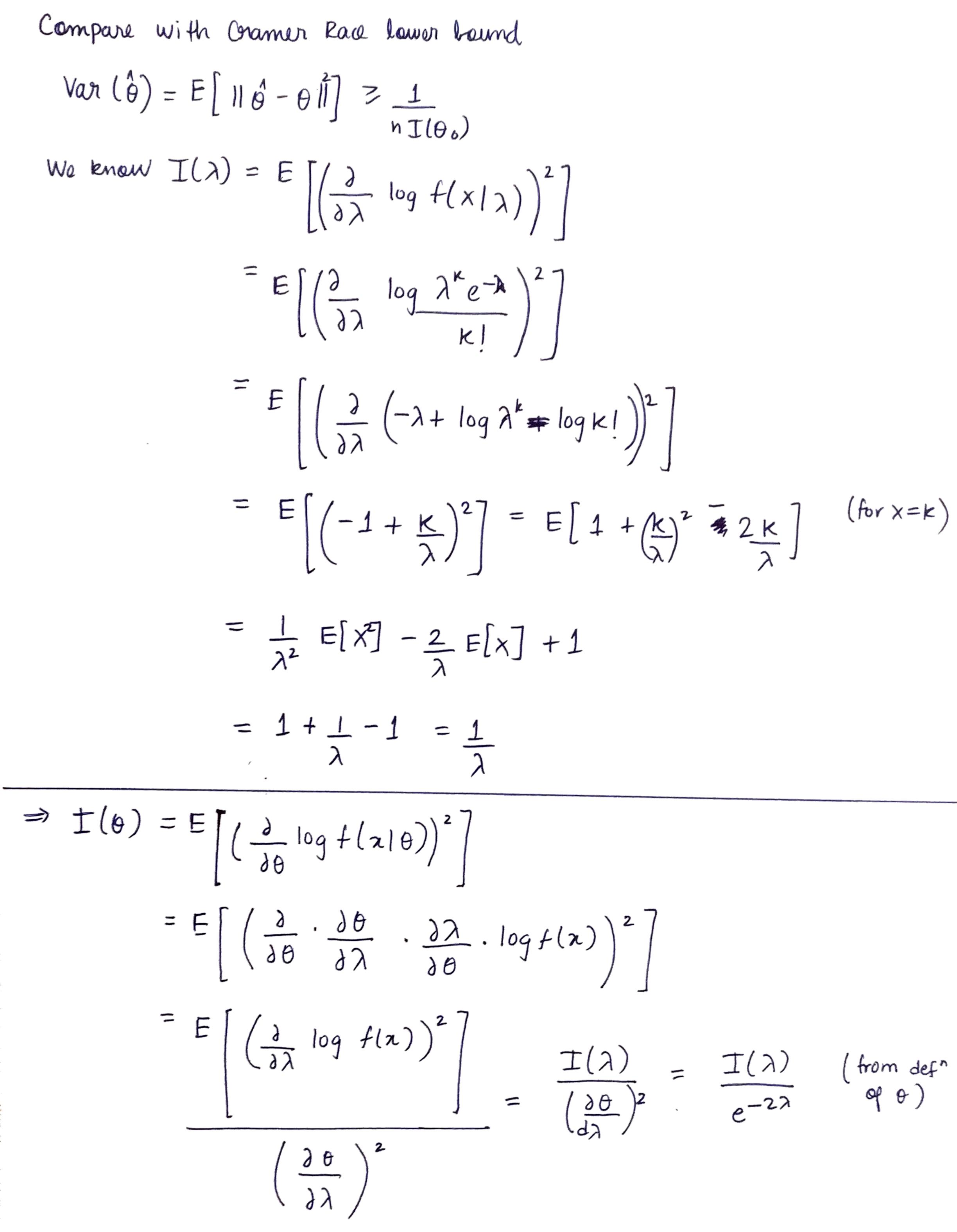
**Problem 4B**

****

****

**Text, letter

Description automatically generated**

****

**A picture containing graphical user interface

Description automatically generated**

**Problem 5A**

**A picture containing text, receipt

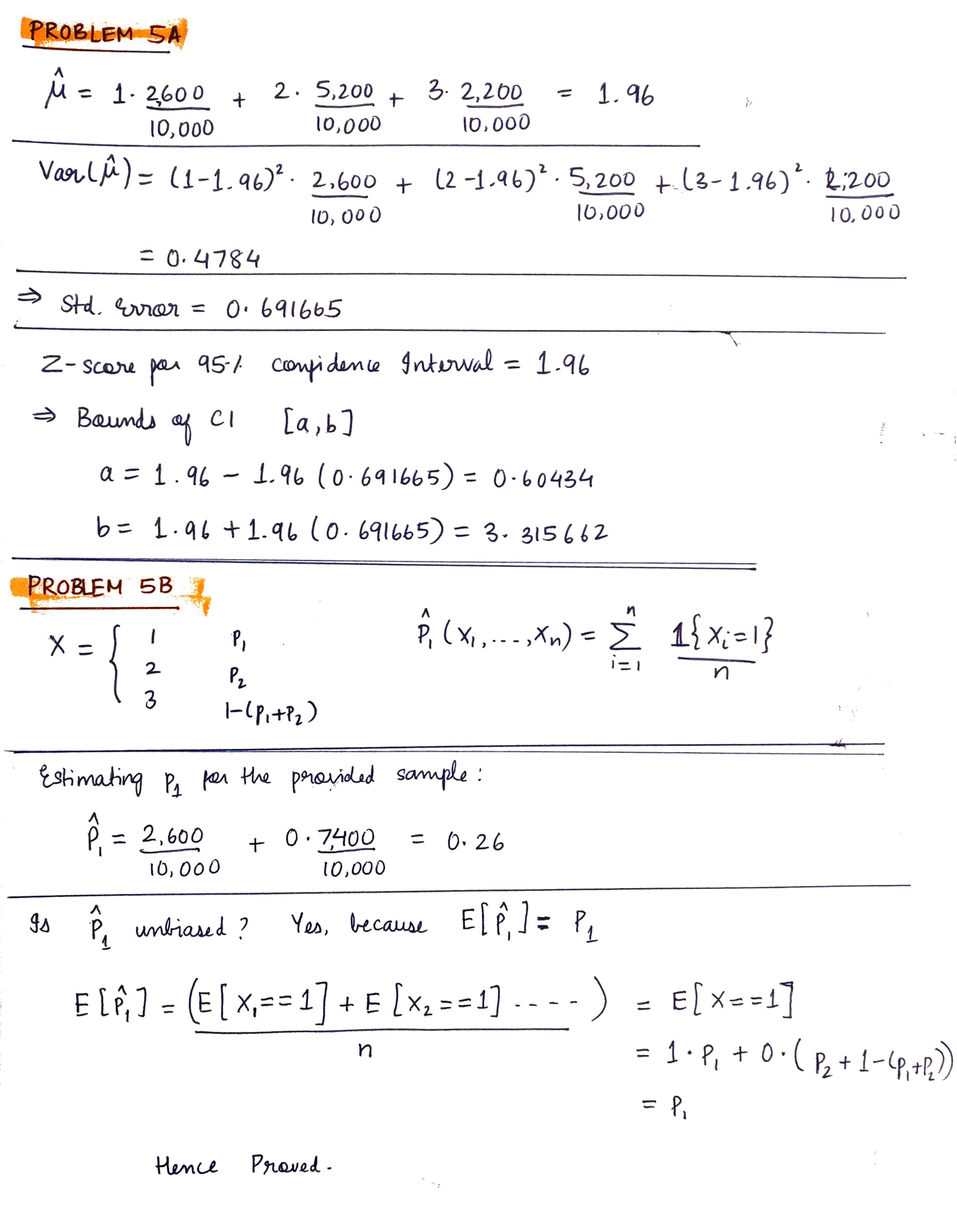
Description automatically generated**

/100 = 1.97

/100 = 1.94

/100

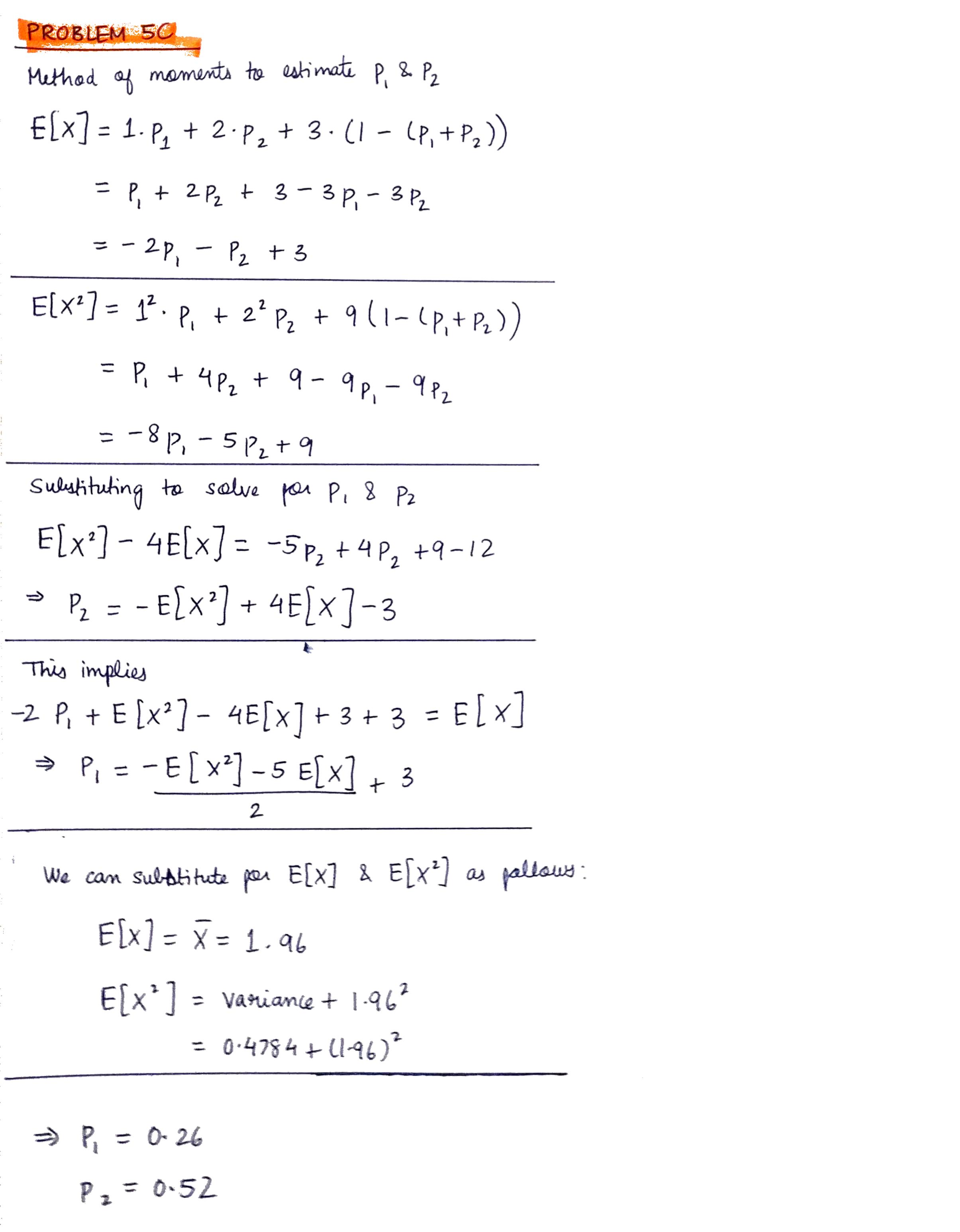
**Problem 5B**

****

**Problem 5C**

**Text, letter

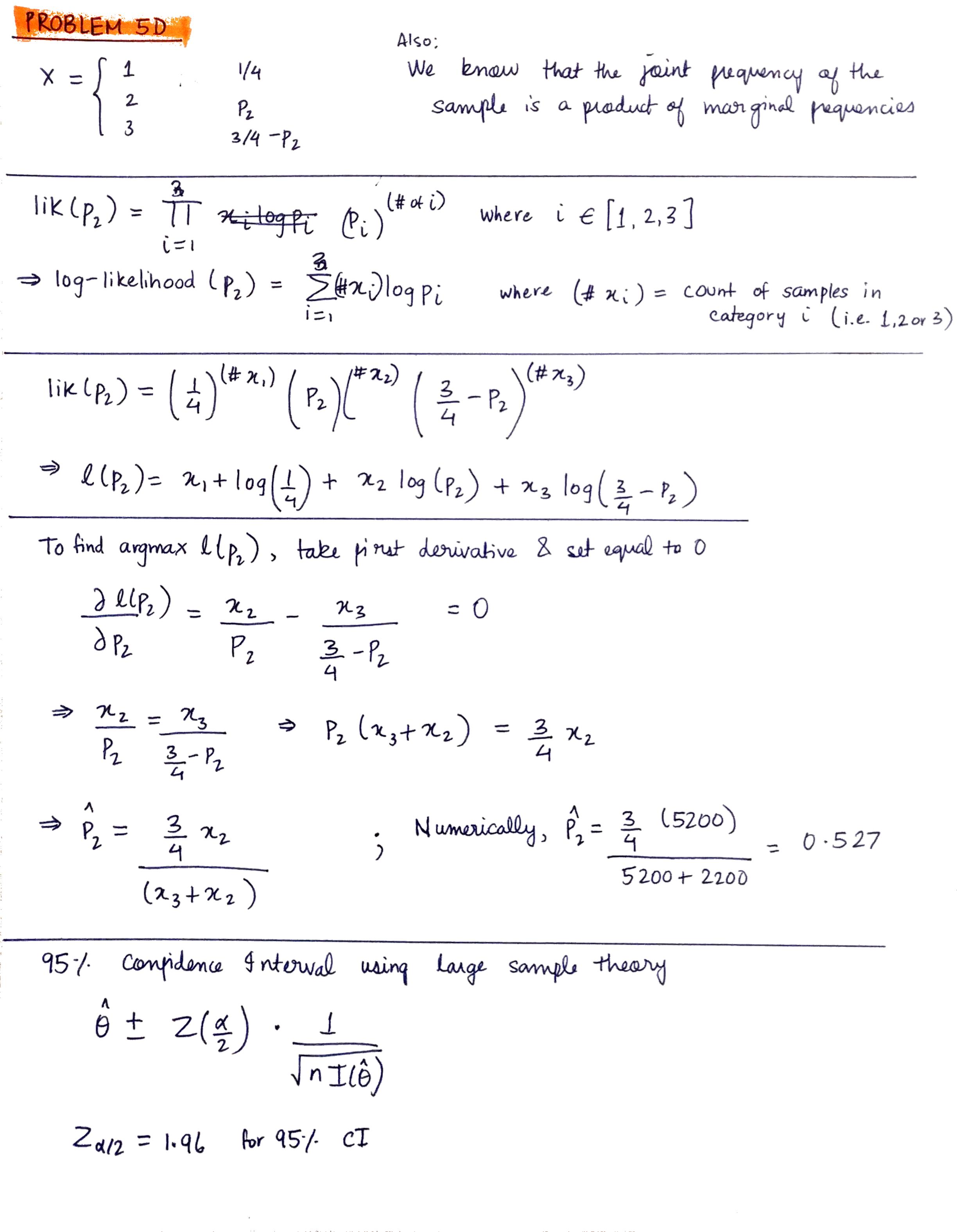
Description automatically generated**

****

**Problem 5D**

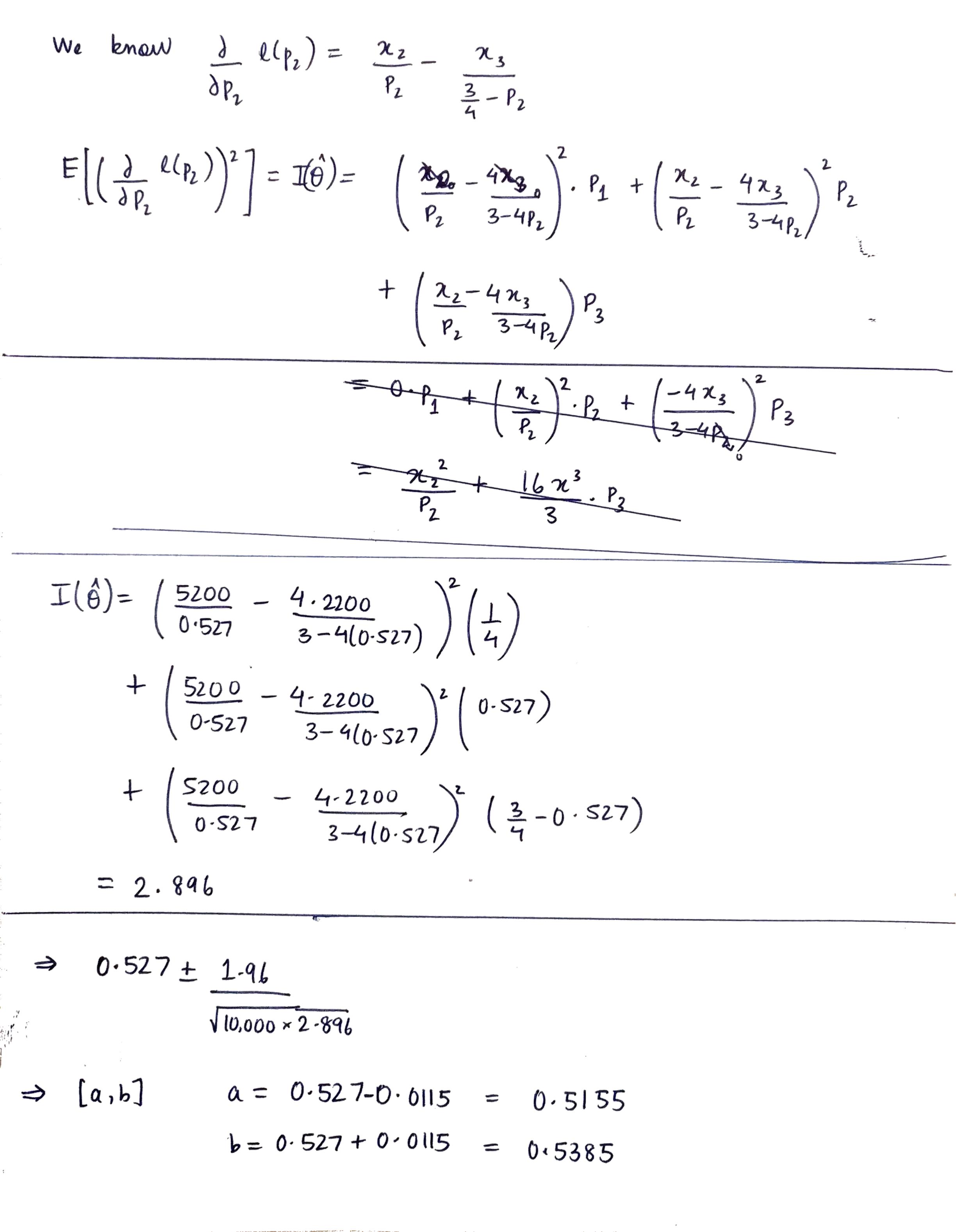
**A piece of paper with writing

Description automatically generated with low confidence**

****

**Text, letter

Description automatically generated**

****

**Problem 6A & 6B**

Graphical user interface, text

Description automatically generated

import csv  
import numpy as np  
import scipy.stats as st  
  
fname = "data\_HW2.csv"  
  
  
def computeMeanWithoutInbuilt(data\_array):  
 sum = 0.0  
 for d in data\_array:  
 sum += d  
 sample\_mean = sum / len(data\_array)  
 return sample\_mean  
  
  
def computeVarianceWithoutInbuilt(data\_array):  
 sample\_mean = computeMeanWithoutInbuilt(data\_array)  
 total\_squared\_deviation = 0.0  
 for d in data\_array:  
 total\_squared\_deviation += (d - sample\_mean) \*\* 2  
 sample\_variance = total\_squared\_deviation / len(data\_array)  
 return sample\_variance  
  
  
with open(fname) as f:  
 reader = csv.reader(f)  
 next(reader) *# skip header* string\_data = [r[0] for r in reader]  
 data\_array = np.asarray(string\_data, dtype=np.float64, order="C")  
  
 sample\_mean\_without\_inbuilt = computeMeanWithoutInbuilt(data\_array)  
 sample\_mean\_with\_inbuilt = np.mean(data\_array)  
 sample\_mean\_deviation = np.abs(sample\_mean\_without\_inbuilt - sample\_mean\_with\_inbuilt)  
  
 sample\_variance\_without\_inbuilt = computeVarianceWithoutInbuilt(data\_array)  
 sample\_variance\_with\_inbuilt = np.nanvar(data\_array)  
 sample\_variance\_deviation = np.abs(sample\_variance\_without\_inbuilt - sample\_variance\_with\_inbuilt)  
  
 ninety\_percent\_confidence\_interval = st.t.interval(alpha=0.90, df=len(data\_array) - 1, loc=np.mean(data\_array),  
 scale=0.25)  
  
 print("Mean computed without inbuilt functions : " + str(sample\_mean\_without\_inbuilt))  
 print("Mean using inbuilt numpy function : " + str(sample\_mean\_with\_inbuilt))  
 print("Deviation between both methods : " + str(sample\_mean\_deviation))  
 print("==========")  
 print("Variance computed without inbuilt functions: " + str(sample\_variance\_without\_inbuilt))  
 print("Variance using inbuilt numpy function : " + str(sample\_variance\_with\_inbuilt))  
 print("Deviation between both methods : " + str(sample\_variance\_deviation))  
 print("==========")  
 print("90% Confidence Interval Bounds : " + str(ninety\_percent\_confidence\_interval))