Lab 4 Problems

Graphs, Multiplicities and Mean Value Theorem

Add needed libraries and variables.

```
# Import necessary libraries
import numpy as np
import sympy as sp
from matplotlib import pyplot as plt

# Define symbols for SymPy
x = sp.symbols('x')
```

Problem 1

Let $f(x) = \frac{x-1}{x^3+2x^2+1}$.

a. Find $f'(x)$. Find and list all real critical points. Leave them in decimal form (you can round them off to the nearest hundredth.)

```
# Define the function
f = (x - 1) / (x**3 + 2*x**2 + 1)

# Compute the derivative of f(x)
f_prime = sp.diff(f, x)
print("Derivative of f(x):", f_prime)

# Solve f'(x) = 0 to find critical points
critical_points = sp.solveset(f_prime, x, domain=sp.S.Reals)

# Convert critical points to decimal form and round to the nearest hundredth
critical_points_decimal = [sp.N(point, 2) for point in critical_points]
print("Critical points (rounded):", critical_points_decimal)
```

```
Derivative of f(x): (x - 1)*(-3*x**2 - 4*x)/(x**3 + 2*x**2 + 1)**2 + 1/(x**3 + 2*x**2 + 1)
Critical points (rounded): [-1.0, -0.28, 1.8]
```

Answer: Critical points (rounded): [-1.0, -0.28, 1.8]

b. Find $f''(x)$. Find all real solutions to $f''(x) = 0$. Leave them in decimal form (you can round them off to the nearest hundredth.)

```
# Compute the second derivative of f(x)
f_double_prime = sp.diff(f_prime, x)
print("Second derivative of f(x):", f_double_prime)

# Solve f''(x) = 0 to find solutions
inflection_points = sp.solveset(f_double_prime, x, domain=sp.S.Reals)

# Convert solutions to decimal form and round to the nearest hundredth
inflection_points_decimal = [sp.N(point, 2) for point in inflection_points]
print("Solutions to f''(x) = 0 (rounded):", inflection_points_decimal)
```

```
Second derivative of f(x): (-6*x - 4)*(x - 1)/(x**3 + 2*x**2 + 1)**2 + (x - 1)*(-6*x**2 - 8*x)*(-3*x**2 - 4*x)/(x**3 + 2*x**2 + 1)**3 + 2*(-3*x**2 - 4*x)/(x**3 + 2*x**2 + 1)**2
Solutions to f''(x) = 0 (rounded): [-0.56, 0.25, 2.5]
```

Answer: Solutions to f''(x) = 0 (rounded): [-0.56, 0.25, 2.5]

c. Find horizontal asymptote if any, by setting up correct limit.

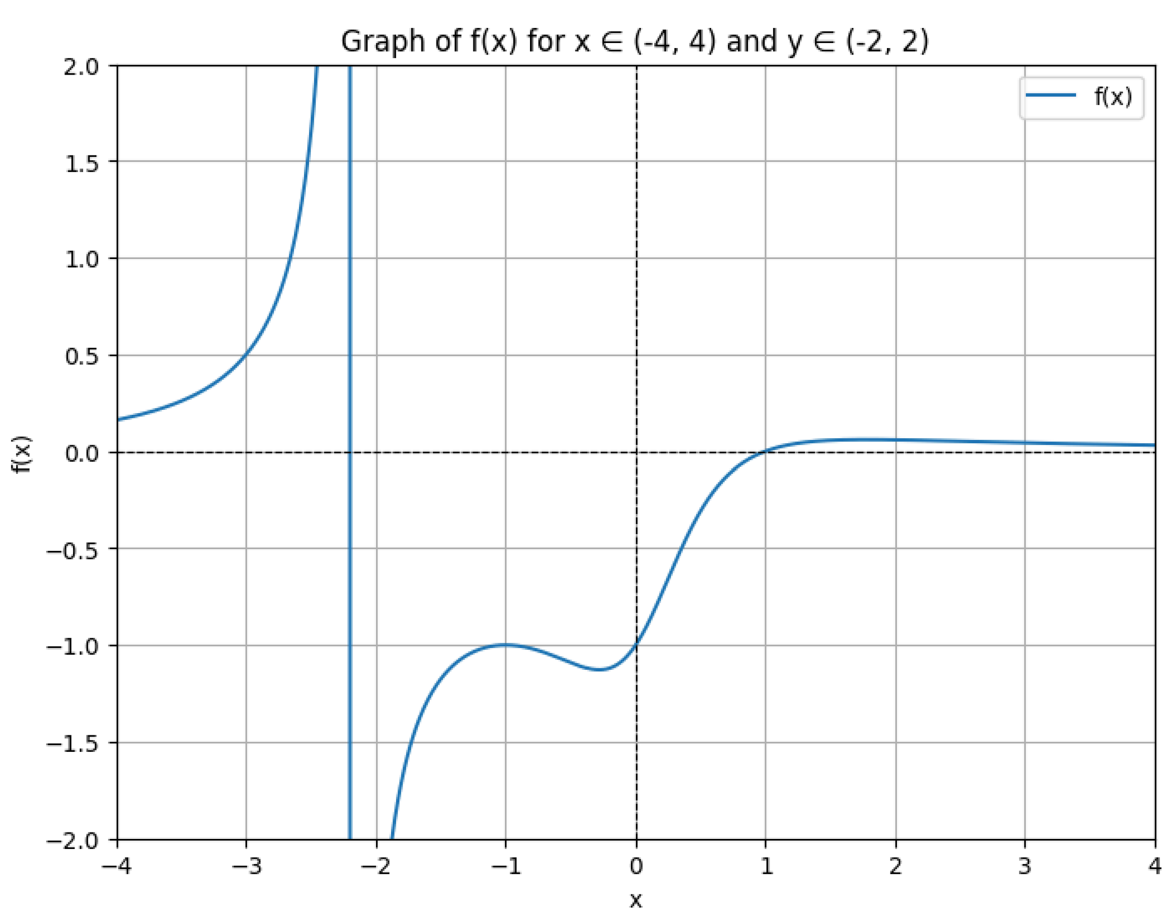
```
# Compute the limit of f(x) as x approaches infinity
horizontal_asymptote_positive = sp.limit(f, x, sp.oo)
print("Horizontal asymptote as x -> ∞:", horizontal_asymptote_positive)

# Compute the limit of f(x) as x approaches negative infinity
horizontal_asymptote_negative = sp.limit(f, x, -sp.oo)
print("Horizontal asymptote as x -> -∞:", horizontal_asymptote_negative)
```

```
Horizontal asymptote as x -> ∞: 0
Horizontal asymptote as x -> -∞: 0
```

Answer: Horizontal asymptote as x -> ∞: 0 Horizontal asymptote as x -> -∞: 0

d. Plot $f(x)$ for $x \in (-4, 4)$ and $y \in (-2, 2)$. Based on results of part **a.** and **b.**, and the graph, list relative maximums, minimums and inflection points.

```
# Import libraries
import numpy as np
import matplotlib.pyplot as plt
import sympy as sp

# Define f(x) as a Python function
f_lambdified = sp.lambdify(x, f, 'numpy')

# Define the range for x and y
x_vals = np.linspace(-4, 4, 1000)
y_vals = f_lambdified(x_vals)

# Plot f(x)
plt.figure(figsize=(8, 6))
plt.plot(x_vals, y_vals, label="f(x)")
plt.axhline(0, color='black', linewidth=0.8, linestyle='--')  # x-axis
plt.axvline(0, color='black', linewidth=0.8, linestyle='--')  # y-axis
plt.xlim(-4, 4)
plt.ylim(-2, 2)
plt.title("Graph of f(x) for x ∈ (-4, 4) and y ∈ (-2, 2)")
plt.xlabel("x")
plt.ylabel("f(x)")
plt.legend()
plt.grid()

# Show the plot
plt.show()
```



Answer:

Maximums: -1, 1.8

Minimums: -0.28

Inflection points: [-0.56, 0.25, 2.5]

Problem 2

Let $f(x) = \frac{2x-1}{\sqrt{4x^2+x+1}}$.

Repeat the work of problem 1 for $f(x)$. When graphing, plot for $x \in (-10, 10)$.

Double-click (or enter) to edit

```
# Define the variable
x = sp.symbols('x')

# Define the function f(x)
f = (2*x - 1) / sp.sqrt(4*x**2 + x + 1)

# Step 1: Compute the first derivative f'(x)
f_prime = sp.diff(f, x)
print("First derivative f'(x):", f_prime)
```

```
First derivative f'(x): (-4*x - 1/2)*(2*x - 1)/(4*x**2 + x + 1)**(3/2) + 2/sqrt(4*x**2 + x + 1)
```

```
# Step 2: Compute the second derivative f''(x)
f_double_prime = sp.diff(f_prime, x)
print("Second derivative f''(x):", f_double_prime)
```

```
Second derivative f''(x): (-12*x - 3/2)*(-4*x - 1/2)*(2*x - 1)/(4*x**2 + x + 1)**(5/2) + 4*(-4*x - 1/2)/(4*x**2 + x + 1)**(3/2) - 4*(2*x - 1)/(4*x**2 + x + 1)**(3/2)
```

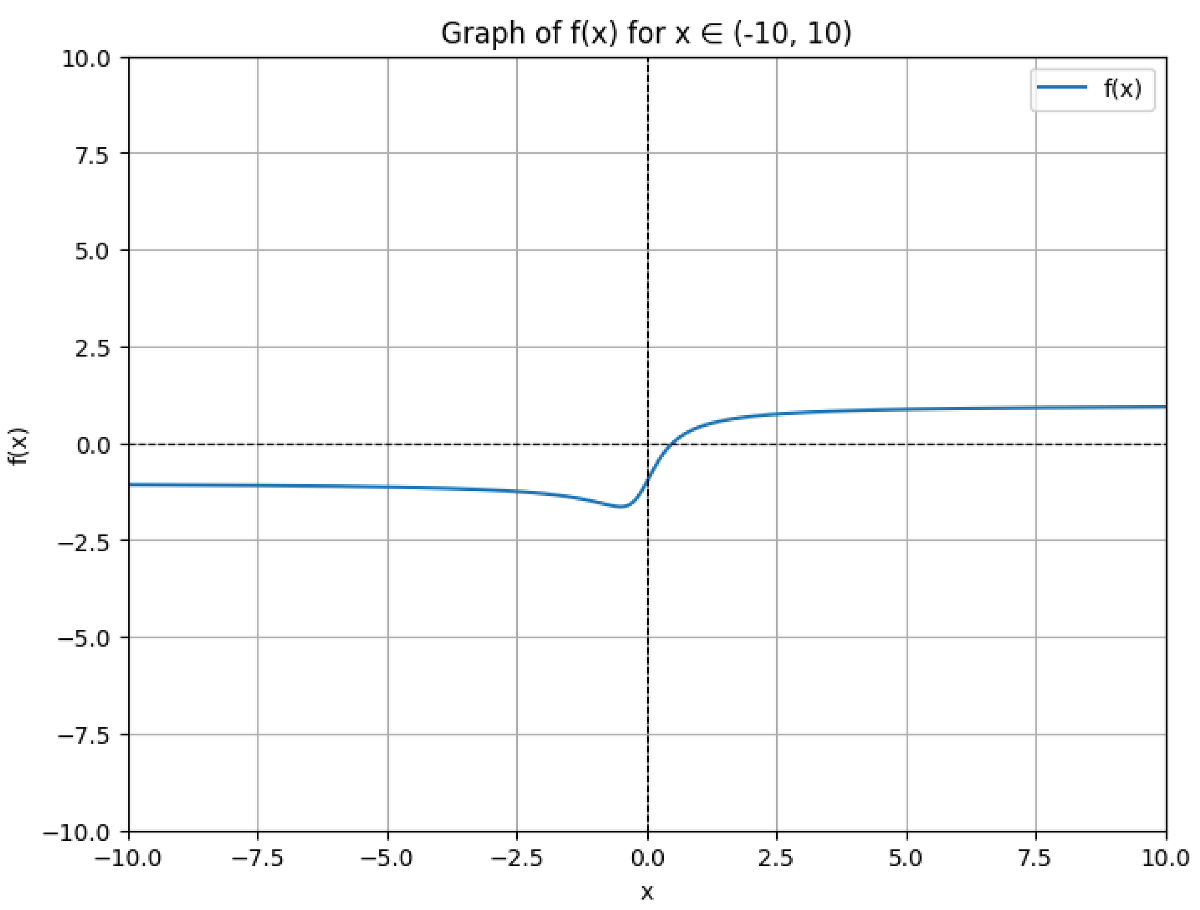
Answer:

```
# Step 3: Find the horizontal asymptote by computing the limit as x -> infinity and x -> -infinity
horizontal_asymptote_pos = sp.limit(f, x, sp.oo)
horizontal_asymptote_neg = sp.limit(f, x, -sp.oo)
print("Limit as x -> ∞:", horizontal_asymptote_pos)
print("Limit as x -> -∞:", horizontal_asymptote_neg)
```

```
Limit as x -> ∞: 1
Limit as x -> -∞: -1
```

Answer:

```
# Step 4: Plot the graph for x ∈ (-10, 10)
f_lambdified = sp.lambdify(x, f, 'numpy')
x_vals = np.linspace(-10, 10, 1000)
y_vals = f_lambdified(x_vals)

plt.figure(figsize=(8, 6))
plt.plot(x_vals, y_vals, label="f(x)")
plt.axhline(0, color='black', linewidth=0.8, linestyle='--')  # x-axis
plt.axvline(0, color='black', linewidth=0.8, linestyle='--')  # y-axis
plt.xlim(-10, 10)
plt.ylim(-10, 10)
plt.title("Graph of f(x) for x ∈ (-10, 10)")
plt.xlabel("x")
plt.ylabel("f(x)")
plt.legend()
plt.grid()
plt.show()

# Now, solve for critical points and inflection points
critical_points = sp.solveset(f_prime, x, domain=sp.S.Reals)
inflection_points = sp.solveset(f_double_prime, x, domain=sp.S.Reals)

# Convert solutions to decimal form and print
critical_points_decimal = [sp.N(point, 2) for point in critical_points]
inflection_points_decimal = [sp.N(point, 2) for point in inflection_points]

print("Relative minimums:", critical_points_decimal)
print("Inflection Points:", inflection_points_decimal)
```




```
Relative minimums: [-0.50]
Inflection Points: [0.037, -0.85]
```

Answer:

Maximums: NaN

Minimums: [-0.50]

Inflection points: [0.037, -0.85]

Problem 3

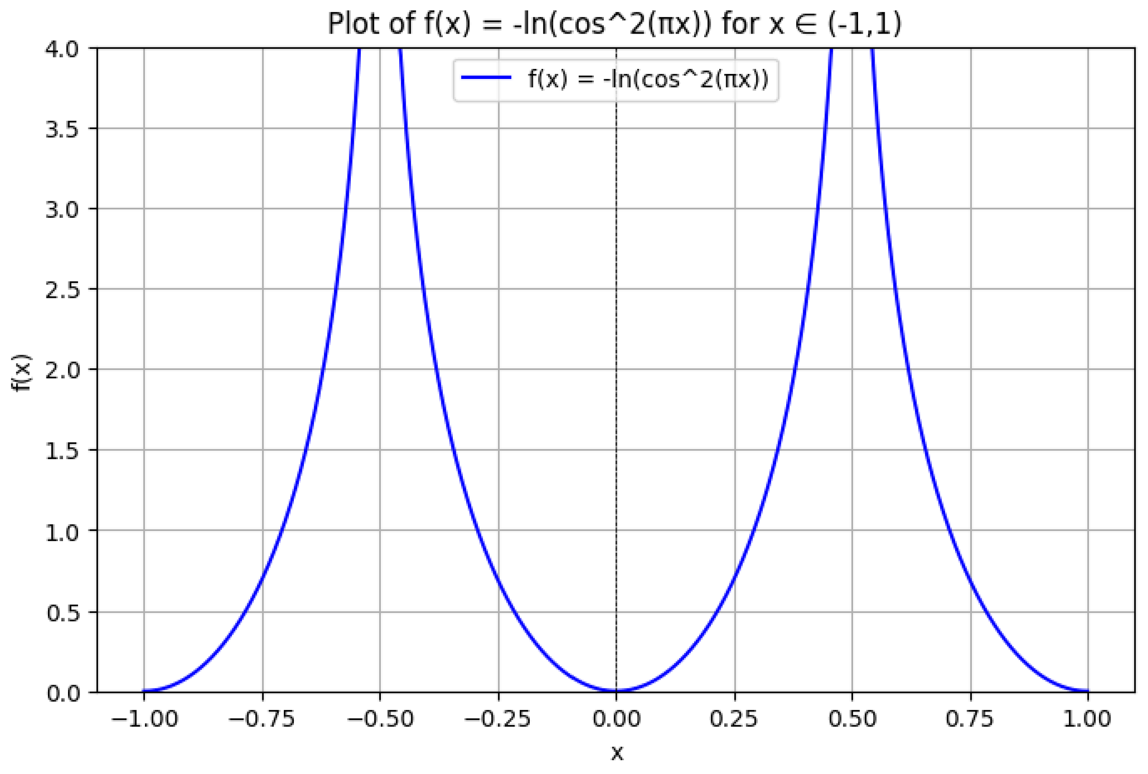
Let $f(x) = -\ln(\cos^2(\pi x))$.

a. Plot $f(x)$ for $x \in (-1, 1)$. Limit y values to interval $(0, 4)$.

Are there any discontinuities in the interval $(-1, 1)$?

Can we apply Mean Value Theorem on the interval $[0, 0.25]$?

```
def f(x):
    return -np.log(np.cos(np.pi * x)**2)

# Define the range for x and calculate y values
x = np.linspace(-1, 1, 1000)
y = f(x)

# Filter y values to limit them to the interval (0, 4)
y_filtered = np.clip(y, 0, 4)

# Plot the function
plt.figure(figsize=(8, 5))
plt.plot(x, y_filtered, label="f(x) = -ln(cos^2(πx))", color='blue')
plt.ylim(0, 4)
plt.axhline(0, color='black', linewidth=0.5, linestyle='--')
plt.axvline(0, color='black', linewidth=0.5, linestyle='--')
plt.title("Plot of f(x) = -ln(cos^2(πx)) for x ∈ (-1,1)")
plt.xlabel("x")
plt.ylabel("f(x)")
plt.legend()
plt.grid()
plt.show()
```



Answer:

b. Find equation of the secant line between $x = 0$ and $x = 0.25$.

```
# Calculate the values of f(x) at x=0 and x=0.25
x1, x2 = 0, 0.25
y1, y2 = f(x1), f(x2)

# Calculate the slope of the secant line
slope = (y2 - y1) / (x2 - x1)

# Calculate the equation of the secant line in slope-intercept form: y = mx + c
intercept = y1 - slope * x1

# Secant line equation: y = slope * x + intercept
secant_line = lambda x: slope * x + intercept

# Output the equation
slope, intercept
```

```
(2.7725887222397803, -0.0)
```

Answer: The equation of the secant line between x=0 and x=0.25 . is, y=2.7726x

c. Find where $f'(x)$ equals to the slope of a secant line on $[0, 0.25]$

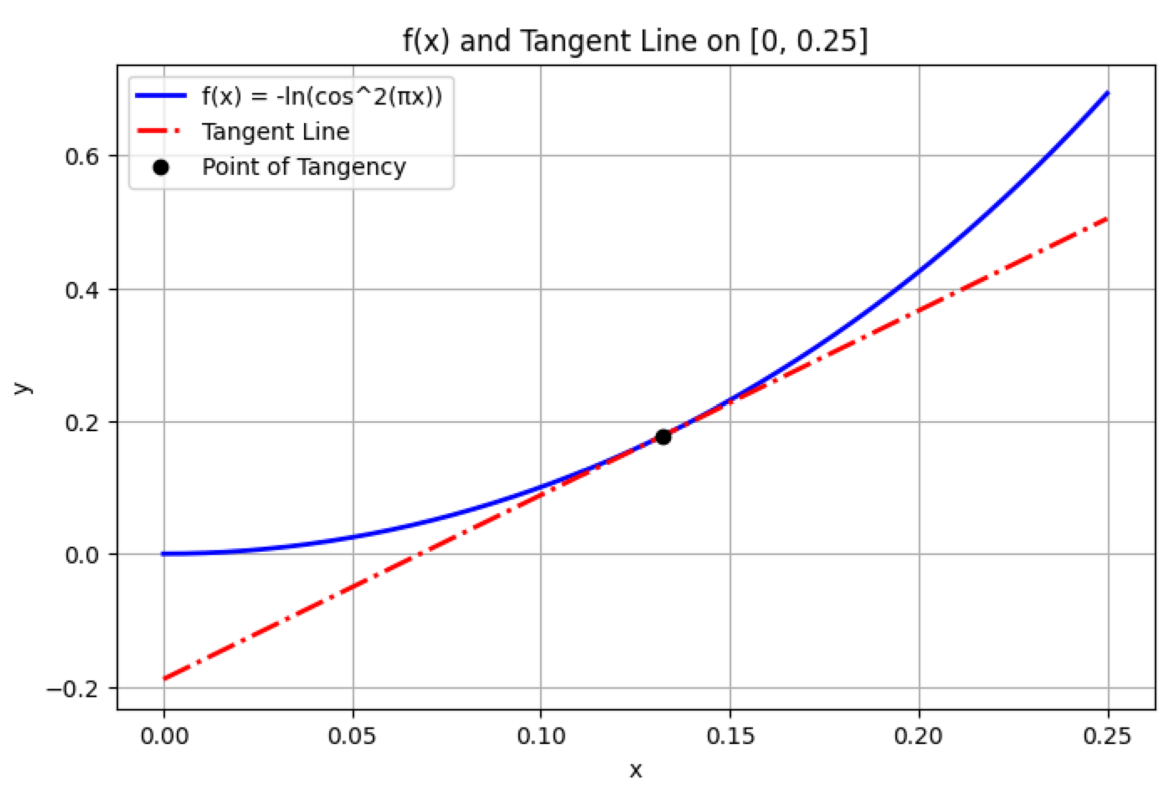
```
from sympy import symbols, diff, cos, pi, log, Eq, solve

# Define the symbolic variable and function
x = symbols('x')
f_sym = -log(cos(pi * x)**2)

# Calculate the derivative of f(x)
f_prime = diff(f_sym, x)

# Set the derivative equal to the slope of the secant line
slope_secant = slope  # from earlier calculation
equation = Eq(f_prime, slope_secant)

# Solve for x in the interval (0, 0.25)
solution = solve(equation, x)

# Filter solutions to the interval (0, 0.25)
valid_solutions = [sol.evalf() for sol in solution if 0 < sol < 0.25]
valid_solutions
```

```
[0.132280481535501]
```

The derivative f ' (x) equals the slope of the secant line (2.7726) at approximately: x=0.1323

d. Find equation of the tangent line at the point found in part **c**. Then graph $f(x)$, secant line and tangent line on the same plot. Use $x \in [0, 0.25]$

```
# Symbolic definition of f(x) and its derivative
x = sp.symbols('x')
f = -sp.ln(sp.cos(sp.pi * x)**2)
f_prime = sp.diff(f, x)

# Convert f and its derivative to numerical functions
f_numeric = sp.lambdify(x, f, modules=["numpy"])
f_prime_numeric = sp.lambdify(x, f_prime, modules=["numpy"])

# Calculate the y-value of f(x) at x_tangent
valid_solutions = [0.1323]  # Example valid solution; replace as needed
x_tangent = valid_solutions[0]
y_tangent = f_numeric(x_tangent)

# Slope of the tangent line is f'(x) at x_tangent
slope_tangent = f_prime_numeric(x_tangent)

# Equation of the tangent line: y = mx + c
intercept_tangent = y_tangent - slope_tangent * x_tangent
tangent_line = lambda x: slope_tangent * x + intercept_tangent

# Define x range for plotting
x_range = np.linspace(0, 0.25, 500)
f_values = f_numeric(x_range)
tangent_values = tangent_line(x_range)

# Plot f(x) and tangent line
plt.figure(figsize=(8, 5))
plt.plot(x_range, f_values, label="f(x) = -ln(cos^2(πx))", color='blue', linewidth=2)
plt.plot(x_range, tangent_values, label="Tangent Line", color='red', linestyle='-.', linewidth=2)
plt.scatter([x_tangent], [y_tangent], color='black', label="Point of Tangency", zorder=5)
plt.title("f(x) and Tangent Line on [0, 0.25]")
plt.xlabel("x")
plt.ylabel("y")
plt.legend()
plt.grid()
plt.show()
```

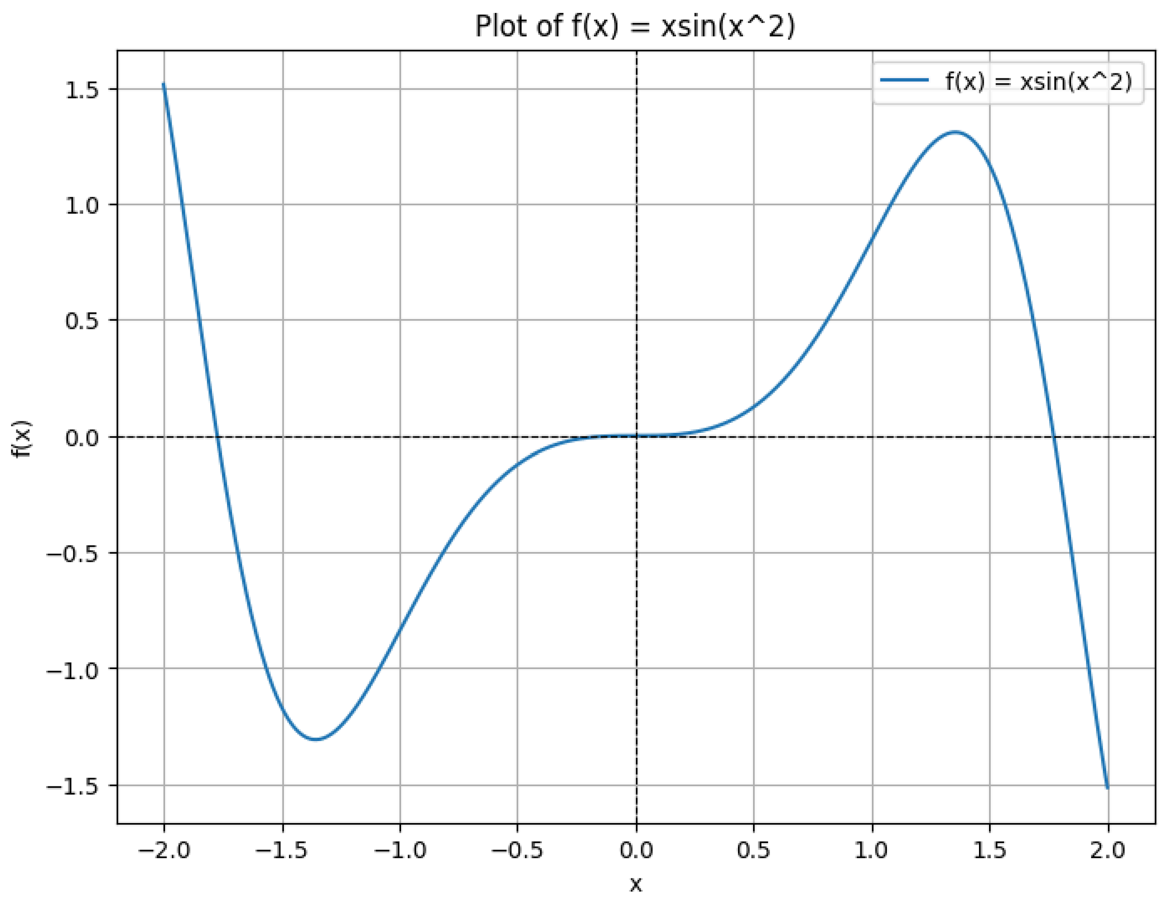


Problem 4

Plot the given functions. Then try to guess the multiplicity of a zero at $x = 0$. Finally, calculate the multiplicity of $x = 0$. (See Examples for definition of multiplicity).

a. $f(x) = x\sin(x^2)$

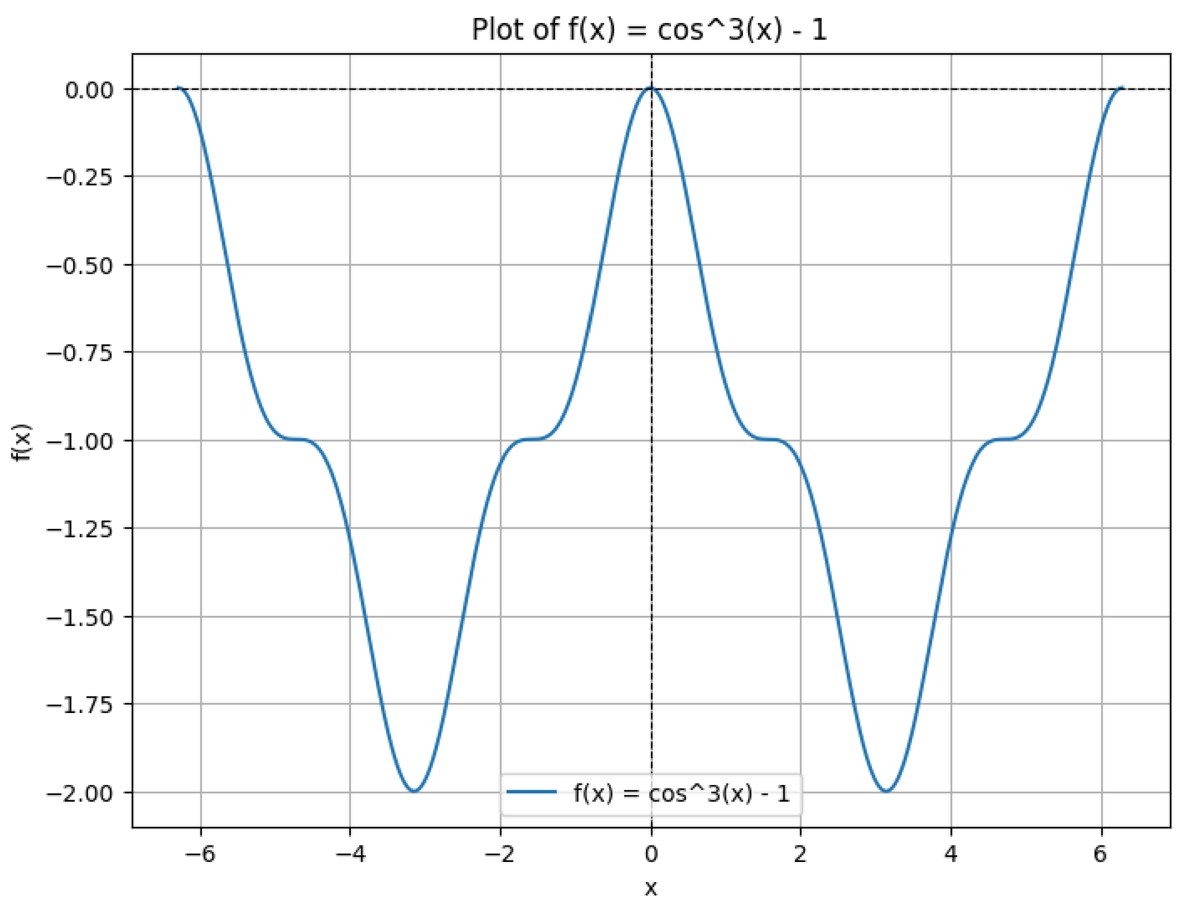
```
def f(x):
    return x * np.sin(x**2)

x_range = np.linspace(-2, 2, 1000)  # Adjust range as needed
y_values = f(x_range)

plt.figure(figsize=(8, 6))
plt.plot(x_range, y_values, label="f(x) = xsin(x^2)")
plt.axhline(0, color='black', linewidth=0.8, linestyle='--')  # x-axis
plt.axvline(0, color='black', linewidth=0.8, linestyle='--')  # y-axis
plt.title("Plot of f(x) = xsin(x^2)")
plt.xlabel("x")
plt.ylabel("f(x)")
plt.legend()
plt.grid(True)
plt.show()
```



Answer: The multiplicity of the zero at x=0 for f(x) = xsin(x^2) is 3.

b. $f(x) = \cos^3(x) - 1$

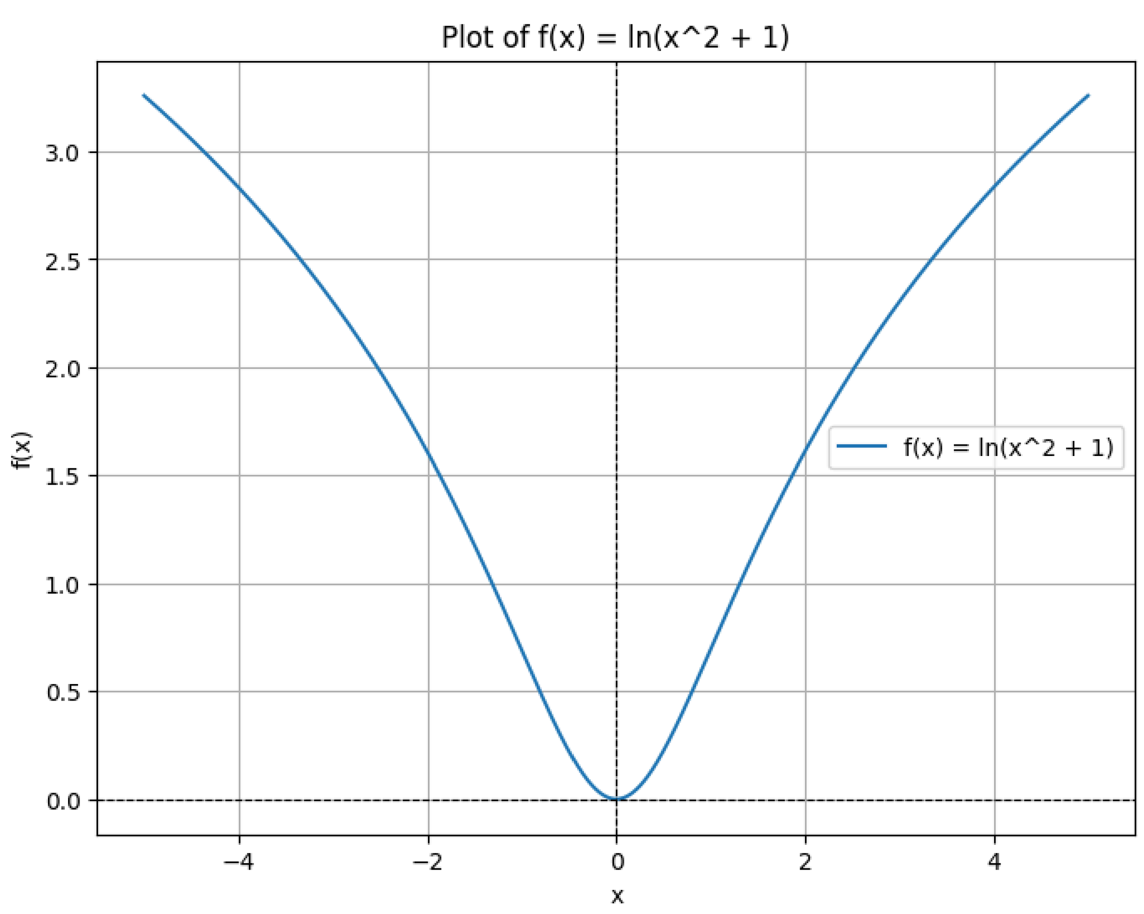
```
def f(x):
    return np.cos(x)**3 - 1

x_range = np.linspace(-2*np.pi, 2*np.pi, 1000)  # Adjust range as needed
y_values = f(x_range)

plt.figure(figsize=(8, 6))
plt.plot(x_range, y_values, label="f(x) = cos^3(x) - 1")
plt.axhline(0, color='black', linewidth=0.8, linestyle='--')  # x-axis
plt.axvline(0, color='black', linewidth=0.8, linestyle='--')  # y-axis
plt.title("Plot of f(x) = cos^3(x) - 1")
plt.xlabel("x")
plt.ylabel("f(x)")
plt.legend()
plt.grid(True)
plt.show()
```



Answer: The multiplicity of the zero at x=0 for f(x) = cos^3(x) - 1 is 2.

c. $f(x) = \ln(x^2 + 1)$

```
def f(x):
    return np.log(x**2 + 1)

x_range = np.linspace(-5, 5, 1000)  # Adjust range as needed
y_values = f(x_range)

plt.figure(figsize=(8, 6))
plt.plot(x_range, y_values, label="f(x) = ln(x^2 + 1)")
plt.axhline(0, color='black', linewidth=0.8, linestyle='--')  # x-axis
plt.axvline(0, color='black', linewidth=0.8, linestyle='--')  # y-axis
plt.title("Plot of f(x) = ln(x^2 + 1)")
plt.xlabel("x")
plt.ylabel("f(x)")
plt.legend()
plt.grid(True)
plt.show()
```

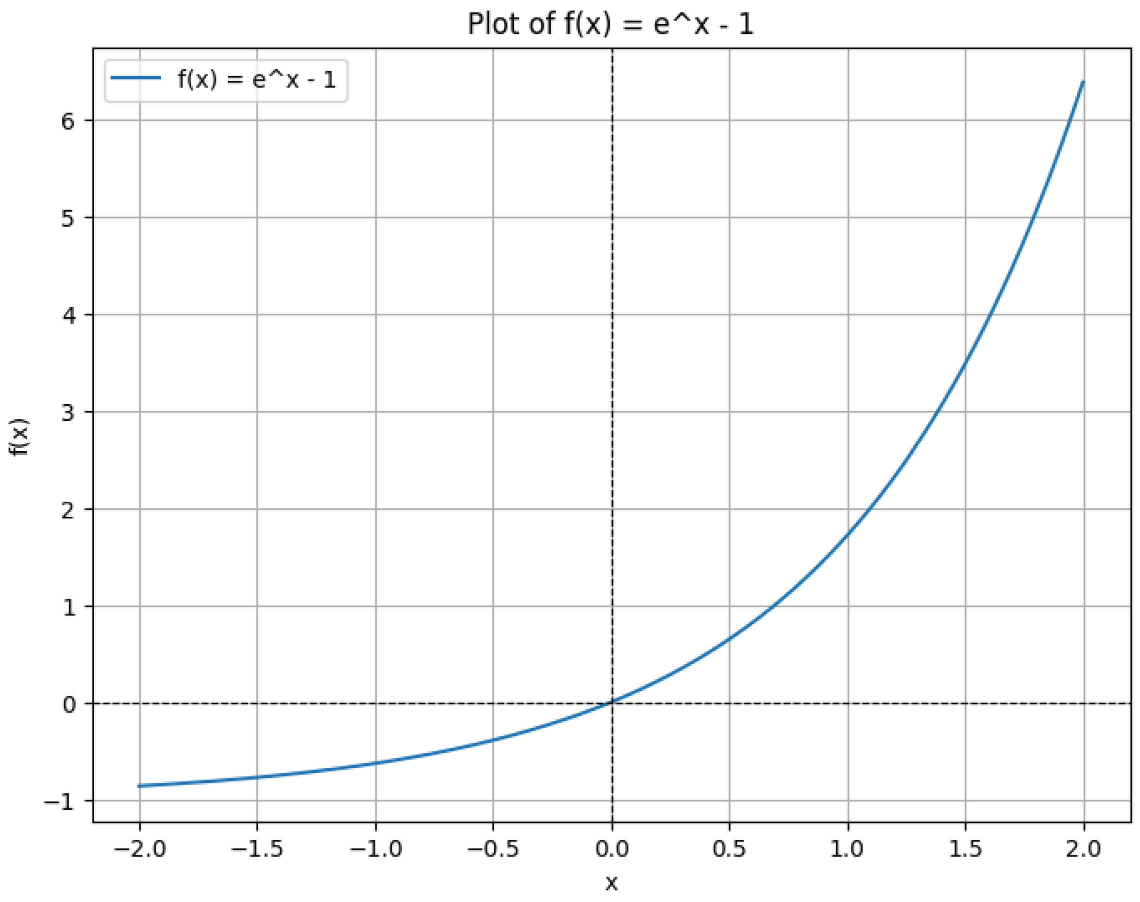


Answer: The multiplicity of the zero at x=0 for f(x) = ln(x^2 + 1) is 2.

d. $f(x) = e^x - 1$

```
def f(x):
    return np.exp(x) - 1
```

```
x_range = np.linspace(-2, 2, 1000)  # Adjust range as needed
y_values = f(x_range)

plt.figure(figsize=(8, 6))
plt.plot(x_range, y_values, label="f(x) = e^x - 1")
plt.axhline(0, color='black', linewidth=0.8, linestyle='--')  # x-axis
plt.axvline(0, color='black', linewidth=0.8, linestyle='--')  # y-axis
plt.title("Plot of f(x) = e^x - 1")
plt.xlabel("x")
plt.ylabel("f(x)")
plt.legend()
plt.grid(True)
plt.show()
```



Answer: The multiplicity of the zero at x=0 for f(x) = e^x - 1 is 1.