Assignment 01

Problem 1

The Fixed Point Iteration Code:

```
'''
Approximates the Root of a Function Using Fixed Point Iteration
g: The Modified Fixed Point Version of the Function
x0: Initial Guess for the Root
tol: Maximum Permissible Error
maxIterations: Maximum Steps to Try Before Giving Up
v: Verbose Mode, Tracks/Returns Intermittent Results
Returns: Approximation of Root, True/False if Iteration Converged, Intermittent
Results if Toggled
'''
def fixedPointIteration(g, x0, tol, maxIterations=100, v=False):
    if v:
        resultsTable = []

    try:
        for i in range(maxIterations):
            x1 = g(x0)
            error = math.fabs(x1 - x0)
            if v:
                resultsTable.append([i, x1, error])

            if error < tol:
                break
            x0 = x1

    except(OverflowError):
        return math.inf, False

    print(error, tol)
    converged = True if error < tol else False

    if v:
        return x1,  converged, resultsTable

    return x1, converged
```

The Code to Run the Problem:

```
def problem1():
    # Original Function
    f = lambda x: x * math.e ** x

    # Fixed Point Alterations
    g1 = lambda x: x - f(x)
    g2 = lambda x: x + f(x)
```

```
roots1 = []
roots2 = []


# Tests Initial Points from -5 to 5 for the First Fixed Point Alteration
print("Approximating the Roots of f(x) = x * e ** x Using g(x) = x - f(x):")
print("----------------------------------------------------------------------")
for i in range(-5, 6):
    print(f"    Using Initial Guess x0 = {i}")
    approximation, converges = fixedPointIteration(g1, i, 0.00000001)
    if converges:
        print(f"        SUCCESS. Result: {approximation}")
        roots1.append(approximation)
    else:
        print(f"        FAILED. Result: {approximation}")

print("Roots Found:")
print("------------")
for root in roots1:
    print(f"x={root}")
print("\n\n\n\n")


# Tests Initial Points from -5 to 5 for the Second Fixed Point Alteration
print("Approximating the Roots of f(x) = x * e ** x Using g(x) = x + f(x):")
print("----------------------------------------------------------------------")

for i in range(-5, 6):
    print(f"    Using Initial Guess x0 = {i}")
    approximation, converges = fixedPointIteration(g2, i, 0.000000001)
    if converges:
        print(f"        SUCCESS. Result: {approximation}")
        roots2.append(approximation)
    else:
        print(f"        FAILED. Result: {approximation}")

print("Roots Found:")
print("------------")
for root in roots2:
    print(f"x={root}")
print("\n\n\n\n")
```

FUNCTION ROOTS: x=0.

This code tests both g(x) given on the interval [-5, 5] in increments of 1 for convergence, and prints the results.

If you run it, you can see a lot more results than you'd expect, as the function has only one root at zero.
However, all the roots it finds are either arbitrarily close to zero, in which case it makes sense as they are within error, or way off in the negative axis.
This makes sense as well, because if you graph the function, you can see that it approaches the x axis as you go left. This means that our algorithm will find "roots" there, because at a certain precision there's no difference between touching and almost touching the x-axis.

Problem 2

The Fixed Point Iteration Code:

```
Same as in Problem 1
```

Problem 3

The Fixed Point Iteration Code:

```
Same as in Problem 1
```

The Code to Run the Problem:

```
def problem3():
    # Original Function
    f = lambda x: 10.14 * math.e ** (x * x) * math.cos(math.pi / x)

    # Fixed Point Alteration
    g = lambda x: x - f(x)

    roots = []

    # Tests Initial Points from -3 to 7 for the Fixed Point Alteration
    print("Approximating the Roots of f(x) = 10.14 * e ** (x * x) * math.cos(pi /
x) Using g(x) = x - f(x):")
    print("----------------------------------------------------------------------
-------------------------")
    for i in range(-3, 8):
        if i == 0:
            i += 0.000001
        print(f"    Using Initial Guess x0 = {i}")
        approximation, converges = fixedPointIteration(g, i, 0.00000001)
        if converges:
            print(f"        SUCCESS. Result: {approximation}")
            roots.append(approximation)
        else:
            print(f"        FAILED. Result: {approximation}")

    print("Roots Found:")
    print("------------")
    for root in roots:
        print(f"x={root}")
    print("\n\n\n\n")
```

FUNCTION ROOTS: x=-2, x=2

This code goes over the interval [-3, 7] and tests for the convergence of g(x) = x - f(x) on each integer step, and prints the results.

Problem 4

The Bisection Code:

```
'''
Approximates the Root of a Function Using the Bisection Method
f: The Function in Question
a: Start of the Interval Containing at Least One Root
b: End of the Interval Containing at Least One Root
tol: Maximum Permissible Error
-v: Verbose Mode, Tracks/Returns Intermittent Results
Returns: The Approximate Root, Intermittent Results if Toggled
NOTES: f(a) and f(b) Must be On Opposite Sides of the x Axis
'''
def bisect(f, a, b, tol, v=False):
    if f(a) * f(b) >= 0:
        raise Exception("ERROR: f(a) and f(b) Must be On Opposite Sides of the x
Axis")

    iterations = math.ceil(math.log2((b - a) / tol))
    print(iterations)
    fa = f(a)
    fb = f(b)
    if v:
        resultsTable = []

    for i in range(iterations):
        c = (a + b) / 2
        fc = f(c)

        if v:
            resultsTable.append([i + 1, c, math.fabs(b - c)])

        if fa * fc < 0:
            fb = fc
            b = c
        elif fb * fc <0 :
            fa = fc
            a = c
        elif fc == 0:
            break
        else:
            raise Exception("ERROR: f(a) and f(b) Must be On Opposite Sides of
the x Axis")

    if v:
        return c, resultsTable

    return c
```

Problem 5

My GitHub link is:

```
https://github.com/jfitzusu/math4610
```

I sent you an invite.