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

    # Forbidden Python Syntax
    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 f(x)
    f = lambda x: x * math.e ** x

    # g(x) Modifications for Fixed Point Iteration
    g1 = lambda x: x - f(x)
    g2 = lambda x: x + f(x)
```

```
    # Tests g1 for Convergence on [-5, 5] in Intervals of 1
    print("Approximating the Roots of f(x) = x * e ** x Using g(x) = x - f(x):")
    for i in range(-5, 6):
        print(f"    Using Initial Guess x0 = {i}")
        approximation, converges = fixedPointIteration(g1, i, 0.00000001)
        if converges:
            print(f"        Convergence SUCCESS. Result: {approximation}")
        else:
            print("        Convergence FAILED")

    # Tests g1 for Convergence on [-5, 5] in Intervals of 1
    print("Approximating the Roots of f(x) = x * e ** x Using g(x) = x + f(x):")
    for i in range(-5, 6):
        print(f"    Using Initial Guess x0 = {i}")
        approximation, converges = fixedPointIteration(g2, i, 0.000000001)
        if converges:
            print(f"        Convergence SUCCESS. Result: {approximation}")
        else:
            print(f"        Convergence FAILED. Result: {approximation}")
```

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

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():
    # The Original f(x)
    f = lambda x: 10.14 * math.e ** (x * x) * math.cos(math.pi / x)

    # The Modified g(x) for FPI
    g = lambda x: x - f(x)
```

```
    print("Approximating the Roots of f(x) = 10.14 * e ** (x * x) * math.cos(pi /
x) Using g(x) = x - f(x):")
    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"        Convergence SUCCESS. Result: {approximation}")
        else:
            print("        Convergence FAILED")
```

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))
    fa = f(a)
    fb = f(b)
    if v:
        resultsTable = []

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

        # Stores the Error for the NEXT Iteration
        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 = b
        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, I'm pretty sure.