Name: Andy Skabelund
Task 1:
The code I wrote for fixed point is:

```
import numpy as np

def g1(x, function):
    gval = x - f_of_x(x)
    return gval

def g2(x, function):
    gval = x + f_of_x(x)
    return gval

def functionalIteration(g, x0, tol, maxIter):
    # g must be a function taking a SINGLE numeric parameter
    function = g
    y0 = x0
    tol = .0000001
    error = 10.0 * tol
    maxiter = 25
    iteration = 0

    while (error > tol and iteration < maxiter):
        x1 = g1(x0, function)
        error = np.abs(x1 - x0)
        x0 = x1
        iteration = iteration + 1

    print("g1 approximation: ", x1)
    error = 10.0 * tol
    iteration = 0

    while (error > tol and iteration < maxiter):
        y1 = g2(y0, function)
        error = np.abs(y1 - y0)
        y0 = y1
        iteration = iteration + 1
    print("g2 approximation: ", y1)


def f_of_x(x):
    return x * np.exp(-x)


functionalIteration(f_of_x, 3, .0000001, 25)
```

Output:

```
g1 approximation:  1.8755956882534777e-26
g2 approximation:  4.836151496037005
```

Task 2:
The code for task 2 is roughly the same except I used Python f strings in order to create a table. The lines I added are:

```
print(f"iter: {iteration}, x0={x0}, g1(x0)={x1}")
print(f"iter: {iteration}, x0={y0}, g2(x0)={y1}")
```

Output:

```
iter: 0, x0=3, g1(x0)=2.8506387948964083
iter: 1, x0=2.8506387948964083, g1(x0)=2.6858508290864664
iter: 2, x0=2.6858508290864664, g1(x0)=2.5027747114002774
iter: 3, x0=2.5027747114002774, g1(x0)=2.297903699973335
iter: 4, x0=2.297903699973335, g1(x0)=2.0670350690292474
iter: 5, x0=2.0670350690292474, g1(x0)=1.8054301412042244
iter: 6, x0=1.8054301412042244, g1(x0)=1.5086107015757624
iter: 7, x0=1.5086107015757624, g1(x0)=1.174880215428721
iter: 8, x0=1.174880215428721, g1(x0)=0.8120114425671736
iter: 9, x0=0.8120114425671736, g1(x0)=0.4515074652923187
iter: 10, x0=0.4515074652923187, g1(x0)=0.1640472578241164
iter: 11, x0=0.1640472578241164, g1(x0)=0.024820036265591566
iter: 12, x0=0.024820036265591566, g1(x0)=0.0006084520637946421
iter: 13, x0=0.0006084520637946421, g1(x0)=3.701013080655317e-07
iter: 14, x0=3.701013080655317e-07, g1(x0)=1.3697495288568338e-13
iter: 15, x0=1.3697495288568338e-13, g1(x0)=1.8755956882534777e-26
g1 approximation:  1.8755956882534777e-26
iter: 0, x0=3, g2(x0)=3.1493612051035917
iter: 1, x0=3.1493612051035917, g2(x0)=3.2844042682874854
iter: 2, x0=3.2844042682874854, g2(x0)=3.4074475647396936
iter: 3, x0=3.4074475647396936, g2(x0)=3.5203214584329268
iter: 4, x0=3.5203214584329268, g2(x0)=3.6244874947785406
iter: 5, x0=3.6244874947785406, g2(x0)=3.7211263300394766
iter: 6, x0=3.7211263300394766, g2(x0)=3.811202473247845
iter: 7, x0=3.811202473247845, g2(x0)=3.8955122264661224
iter: 8, x0=3.8955122264661224, g2(x0)=3.9747195079017277
iter: 9, x0=3.9747195079017277, g2(x0)=4.049382903453073
iter: 10, x0=4.049382903453073, g2(x0)=4.1199763189702345
iter: 11, x0=4.1199763189702345, g2(x0)=4.186904918703634
iter: 12, x0=4.186904918703634, g2(x0)=4.250517555244832
iter: 13, x0=4.250517555244832, g2(x0)=4.311116560436654
iter: 14, x0=4.311116560436654, g2(x0)=4.368965530612663
iter: 15, x0=4.368965530612663, g2(x0)=4.424295572213126
iter: 16, x0=4.424295572213126, g2(x0)=4.477310354187183
```

```
iter: 17, x0=4.477310354187183, g2(x0)=4.528190227217746
iter: 18, x0=4.528190227217746, g2(x0)=4.577095606837029
iter: 19, x0=4.577095606837029, g2(x0)=4.624169771149885
iter: 20, x0=4.624169771149885, g2(x0)=4.669541189441271
iter: 21, x0=4.669541189441271, g2(x0)=4.713325472119624
iter: 22, x0=4.713325472119624, g2(x0)=4.7556270129142755
iter: 23, x0=4.7556270129142755, g2(x0)=4.796540379346254
iter: 24, x0=4.796540379346254, g2(x0)=4.836151496037005
g2 approximation:  4.836151496037005
```

Task 3:
My functional iteration code for task 3 is:

```
import numpy as np

def functionalIteration(g, x0, tol, maxIter):
    # g must be a function taking a SINGLE numeric parameter
    function = g
    y0 = x0
    tol = .0000001
    error = 10.0 * tol
    maxiter = 25
    iteration = 0

    while (error > tol and iteration < maxiter):
        x1 = g(x0)
        error = np.abs(x1 - x0)
        print(f"iter: {iteration}, x0={x0}, fx(x0)={x1}")
        x0 = x1
        iteration = iteration + 1

    print("fx approximation: ", x1)
    error = 10.0 * tol
    iteration = 0

def f_of_x(x):
    return x - 10.14 * np.exp(x * x) * np.cos(np.pi / x)

functionalIteration(f_of_x, 2, .0000001, 25)
```

Output:

```
iter: 0, x0=2, fx(x0)=1.999999999999966
fx approximation:  1.999999999999966
```

Task 4:
The code I wrote for bisection is:

```
import numpy as np
```

```
def f_of_x(x):
    return x - x * np.exp(-x)

def h_of_x(x):
    return x + x * np.exp(-x)

def l_of_x(x):
    return x - 10.14 * np.exp(x * x) * np.cos(np.pi / x)

def bisection1(f, a, b, tol):
    x = a
    fa = f_of_x(a)
    x = b
    fb = f_of_x(b)

    n = int((np.log(tol) - np.log(b - a)) / np.log(.5) + 1)

    for i in range(1, n):
        c = .5 * (a + b)
        x = c
        fc = f_of_x(c)
        if fa * fc < 0:
            b = c
            fb = fc
        else:
            a = c
            fa = fc
    return c

def bisection2(f, a, b, tol):
    x = a
    ha = h_of_x(a)
    x = b
    hb = h_of_x(b)

    n = int((np.log(tol) - np.log(b - a)) / np.log(.5) + 1)

    for i in range(1, n):
        c = .5 * (a + b)
        x = c
        hc = f_of_x(c)
        if ha * hc < 0:
            b = c
            hb = hc
        else:
```

```
            a = c
            ha = hc
    return c

def bisection3(f, a, b, tol):
    x = a
    la = l_of_x(a)
    x = b
    lb = l_of_x(b)

    n = int((np.log(tol) - np.log(b - a)) / np.log(.5) + 1)

    for i in range(1, n):
        c = .5 * (a + b)
        x = c
        lc = l_of_x(c)
        if la * lc < 0:
            b = c
            lb = lc
        else:
            a = c
            la = lc
    return c

print("f(x) approximation: ", bisection1(f_of_x, -1, 2.2, .000001))
print("h(x) approximation: ", bisection2(h_of_x, -1, 2.2, .000001))
print("l(x) approximation: ", bisection3(l_of_x, -1, 2.2, .000001))
```

Output:

```
f(x) approximation:  2.199998474121094
h(x) approximation:  -0.9999984741210938
l(x) approximation:  0.40172576904296875
```

Task 5:
The link to my github is: https://github.com/JetzNation/math4610