(A)

```
import numpy as np
import math

def getInitalVals(a, h):
    return [f(a + 3*h), f(a + 2*h), f(a + h), f(a)]

def f(t):
    return -1 + math.log(2 - math.exp(1 - math.exp(t)))

def f_p(t, y):
    return math.exp(t)*(2*math.exp(-1*(y+1)) - 1)

def AdamsB(h, temp_arr, t):
    w3, w2, w1, w0 = temp_arr

    return w3 + (h/24)*(55*f_p(t + 3*h, w3) - 59*f_p(t + 2*h, w2) +
                    37*f_p(t + h, w1) - 9*f_p(t, w0))

def AdamsM(h, temp_arr, t):
    wp, w3, w2, w1 = temp_arr
    return w3 + (h/24)*(9*f_p(t + 4*h, wp) + 19*f_p(t + 3*h, w3)
                    - 5*f_p(t + 2*h, w2) + f_p(t + h, w1))

def TrueV(a):
    return f(a)
def Error(true, pred):
    return abs(pred - true)
def AdamsPredCorr(a, b, h, p_arr):
    n = int((b - h*3)/h)
    temp_arr = np.zeros((len(p_arr)))
    temp_arr = p_arr
    for i in range(0, n + 1):
        t = h*i
        wp = AdamsB(h, temp_arr, t)
        temp_arr = np.roll(temp_arr, 1)
        temp_arr[0] = wp
        w = AdamsM(h, temp_arr, t)
    return w

def get_h_arr():
    h_arr = np.zeros((7))
    for i in range(3, 10):
        h_arr[i - 3] = 1/2**i
    return h_arr
```

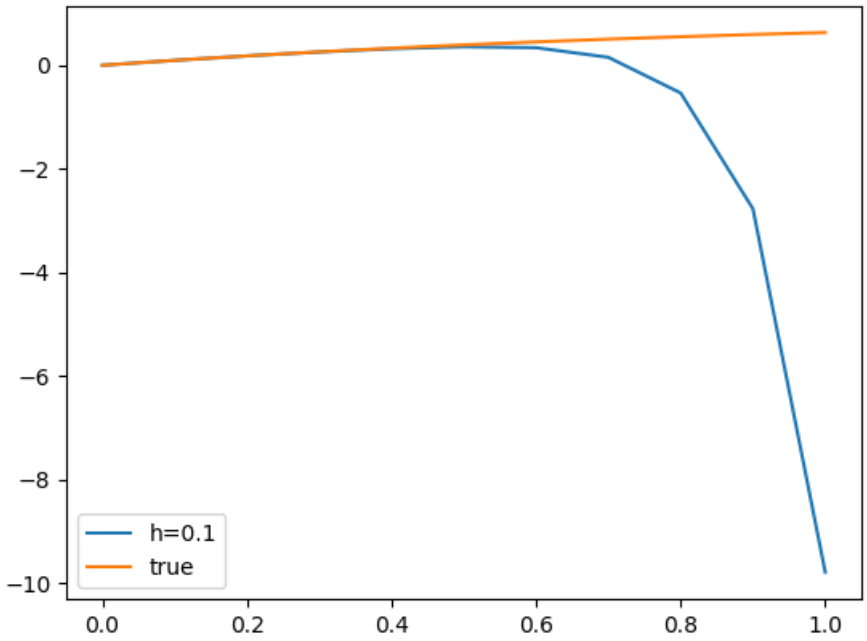
```
def main():
    a = 0
    b = 1
    h_arr = get_h_arr()
    true = TrueV(b)
    for i in range(0, len(h_arr)):
        h = h_arr[i]
        InitVals = getInitalVals(a, h)
        w = AdamsPredCorr(a, b, h, InitVals)
        error = Error(true, w)
        print('h:', h, '|| Predicted:', w, '|| Error:', error)

if __name__ == "__main__":
    main()

Output:
 h: 0.125 || Predicted: -0.37139193019735584 || Error: 0.029427715054879067
 h: 0.0625 || Predicted: -0.38510418960094883 || Error: 0.01571545565128607
 h: 0.03125 || Predicted: -0.3927079985193765 || Error: 0.008111646732858413
 h: 0.015625 || Predicted: -0.3966996945208886 || Error: 0.0041199507313463
 h: 0.0078125 || Predicted: -0.39874354349057844 || Error: 0.002076101761656457
 h: 0.00390625 || Predicted: -0.3997775498994745 || Error: 0.0010420953527603971
 h: 0.001953125 || Predicted: -0.400297584777733 || Error: 0.0005220604745019286
# Notice that Error is directly proportionally to h (1/2 h -> 1/2 Error)
```

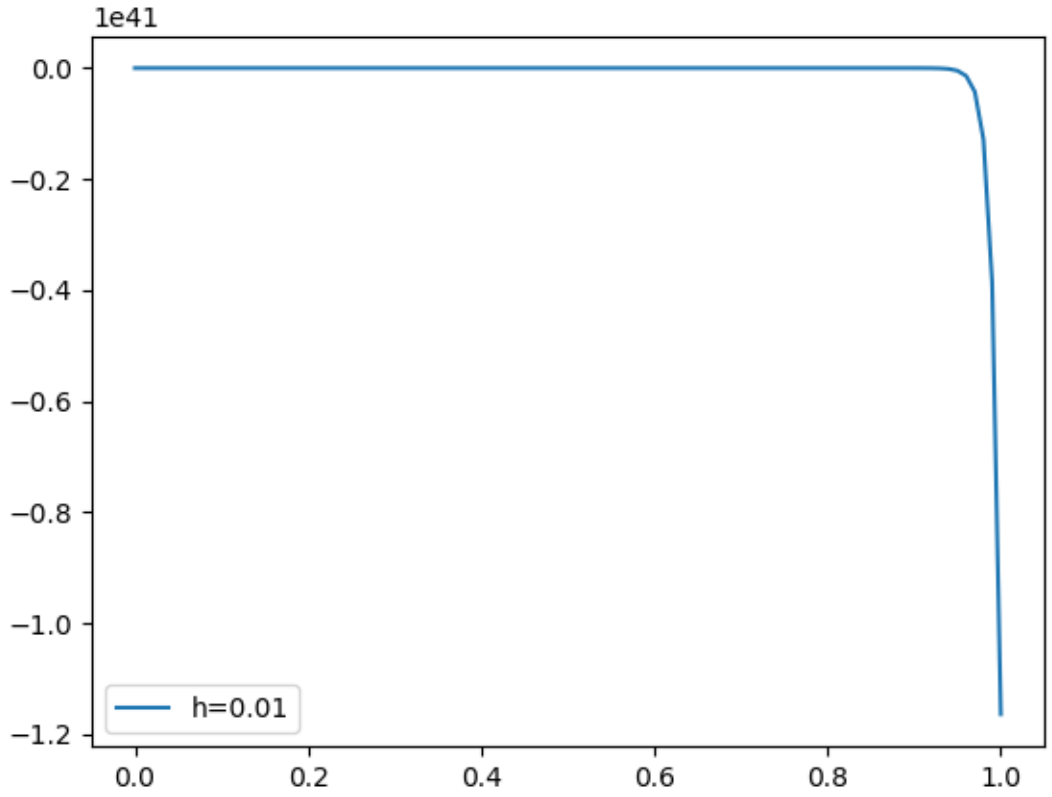
(B) Below in Math work

#4(a)(b)(c)(d)
Below in Math work

```
import numpy as np
import matplotlib.pyplot as plt
import math

def Method(a, b, alp, h):

    # Finding total number of iterations in [a,b]
    N = int((b - a) / h)

    # Initial Condidtions
    w = alp
    w1 = 1 - math.exp(-h)

    # Using difference iteration method
    w_arr = np.zeros((N + 1))
```

```
    t_arr = np.zeros((N + 1))

    w_arr[0] = w
    w_arr[1] = w1
    t_arr[0] = a
    t_arr[1] = a + h
    for i in range(0, N - 1):
        w2 = 4*w1 -3*w - 2*h*(1 - w)
        w_arr[i + 2] = w2
        t_arr[i + 2] = t_arr[i + 1] + h
        w = w1
        w1 = w2
    return w2, w_arr, t_arr
def true_func(t):
    x = np.zeros((len(t)))
    for i in range(0, len(x)):
        x[i] = -math.exp(-t[i]) + 1
    return x
def main():
    a = 0
    b = 1
    alpha = 0
    h = 0.1
    approx, w_arr, t_arr = Method(a, b, alpha, h)
    h = 0.01
    approx2, w_arr2, t_arr2 = Method(a, b, alpha, h)
    true_arr = true_func(t_arr)

    plt.plot(t_arr, w_arr, label='h=0.1')
    plt.plot(t_arr, true_arr, label='true')
    plt.plot(t_arr2, w_arr2, label='h=0.01')
    plt.legend()
    plt.show()
if __name__ == "__main__":
    main()
```


0
−2
−4
−6
−8
−10
0.0
0.2
0.4
0.6
0.8
1.0
h=0.1
true


1e41
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
0.0
0.2
0.4
0.6
0.8
1.0
h=0.01

#2c

```
import numpy as np
import math


def f1(t, u1, u2, u3):
    return u1 + 2*u2 - 2*u3 + math.exp(-t)
def f2(t, u1, u2, u3):
    return u2 + u3 - 2*math.exp(-t)
def f3(t, u1, u2, u3):
    return u1 + 2*u2 + math.exp(-t)

def k11 (h, t, u1, u2, u3):
    return h*f1(t, u1, u2, u3)
def k12 (h, t, u1, u2, u3):
    return h*f2(t, u1, u2, u3)
def k13 (h, t, u1, u2, u3):
    return h*f3(t, u1, u2, u3)

def k21 (t, h, u1, k11x, u2, k12x, u3, k13x):
    return h*f1(t + h/2, u1 + k11x/2, u2 + k12x/2, u3 + k13x/2)
def k22 (t, h, u1, k11x, u2, k12x, u3, k13x):
    return h*f2(t + h/2, u1 + k11x/2, u2 + k12x/2, u3 + k13x/2)
def k23 (t, h, u1, k11x, u2, k12x, u3, k13x):
    return h*f3(t + h/2, u1 + k11x/2, u2 + k12x/2, u3 + k13x/2)

def k31 (t, h, u1, k21x, u2, k22x, u3, k23x):
    return h*f1(t + h/2, u1 + k21x/2, u2 + k22x/2, u3 + k23x/2)
def k32 (t, h, u1, k21x, u2, k22x, u3, k23x):
    return h*f2(t + h/2, u1 + k21x/2, u2 + k22x/2, u3 + k23x/2)
def k33 (t, h, u1, k21x, u2, k22x, u3, k23x):
    return h*f3(t + h/2, u1 + k21x/2, u2 + k22x/2, u3 + k23x/2)

def k41 (t, h, u1, k31x, u2, k32x, u3, k33x):
    return h*f1(t + h, u1 + k31x, u2 + k32x, u3 + k33x)
def k42 (t, h, u1, k31x, u2, k32x, u3, k33x):
    return h*f2(t + h, u1 + k31x, u2 + k32x, u3 + k33x)
def k43 (t, h, u1, k31x, u2, k32x, u3, k33x):
    return h*f3(t + h, u1 + k31x, u2 + k32x, u3 + k33x)

def w1 (u1, k11x, k21x, k31x, k41x):
```

```
    return u1 + (1/6)*(k11x + 2*k21x + 2*k31x + k41x)
def w2 (u2, k12x, k22x, k32x, k42x):
    return u2 + (1/6)*(k12x + 2*k22x + 2*k32x + k42x)
def w3 (u3, k13x, k23x, k33x, k43x):
    return u3 + (1/6)*(k13x + 2*k23x + 2*k33x + k43x)

def u1_true(t):
    return -3*math.exp(-t) - 3*math.sin(t) + 6*math.cos(t)
def u2_true(t):
    return 3*math.exp(-
t)/2 + 0.3*math.sin(t) - 2.1*math.cos(t) - 0.4*math.exp(2*t)
def u3_true(t):
    return -math.exp(-t) +12*math.cos(t)/5 + 9*math.sin(t)/5 - 2*math.exp(2*t)/5

def Error(u1, u2, u3):
    return abs(u1 - u1_true(1)), abs(u2 - u2_true(1)), abs(u3 - u3_true(1))
def RK4_Systems3(u1, u2, u3, a, b, h):
    N = int((b - a) / h)
    for i in range(0, N):
        t = a + i*h

        k11x = k11(h, t, u1, u2, u3)
        k12x = k12(h, t, u1, u2, u3)
        k13x = k13(h, t, u1, u2, u3)

        k21x = k21(t, h, u1, k11x, u2, k12x, u3, k13x)
        k22x = k22(t, h, u1, k11x, u2, k12x, u3, k13x)
        k23x = k23(t, h, u1, k11x, u2, k12x, u3, k13x)

        k31x = k31(t, h, u1, k21x, u2, k22x, u3, k23x)
        k32x = k32(t, h, u1, k21x, u2, k22x, u3, k23x)
        k33x = k33(t, h, u1, k21x, u2, k22x, u3, k23x)

        k41x = k41(t, h, u1, k31x, u2, k32x, u3, k33x)
        k42x = k42(t, h, u1, k31x, u2, k32x, u3, k33x)
        k43x = k43(t, h, u1, k31x, u2, k32x, u3, k33x)

        u1 = w1(u1, k11x, k21x, k31x, k41x)
        u2 = w2(u2, k12x, k22x, k32x, k42x)
        u3 = w3(u3, k13x, k23x, k33x, k43x)

    return u1, u2, u3

def main():
    a = 0
```

```
    b = 1
    u1 = 3
    u2 = -1
    u3 = 1
    h = 0.1

    w1, w2, w3 = RK4_Systems3(u1, u2, u3, a, b, h)
    e1, e2, e3 = Error(w1, w2, w3)
    print('Pred u1:', w1, '|| Error:', e1)
    print('Pred u2:', w2, '|| Error:', e2)
    print('Pred u3:', w3, '|| Error:', e3)


if __name__ == "__main__":
    main()

Output:
Pred u1: -0.3862304827680181 || Error: 6.959961159624939e-06
Pred u2: -3.285940713040603 || Error: 5.611655218682444e-05
Pred u3: -0.5120702580005883 || Error: 5.8316005365322e-05
```

(C) Below in math work

(D) Below in math work