```python
import numpy as np
from scipy import integrate
from mpl_toolkits.mplot3d import Axes3D
from HW1 import convert_rv_kep, kepler_J2_ODE
import matplotlib.pyplot as plt

# constants
params = {'mu_E': 398600.0,
          'J2': 1.08262692563 8815e-03,
          'R_E': 6378.1363,
          'n_objs': 2}

# orbit elems of first satellite
a = 10000.0  # [km]
e = 0.001  # [km]
inc = 40.0  # [deg]
Omega = 80.0  # [deg]
omega = 40.0  # [deg]
M_0 = 0.0  # [deg]
elems_1 = np.array([a, e, inc, Omega, omega, M_0])
# convert to state vector
x_1 = convert_rv_kep.convert_rv_kep('keplerian', elems_1, 0.0, 'state').reshape(6)

# orbit elems of second satellite
a = 10000.0  # [km]
e = 0.8  # [km]
inc = 90.0  # [deg]
Omega = 80.0  # [deg]
omega = 40.0  # [deg]
M_0 = 0.0  # [deg]
elems_2 = np.array([a, e, inc, Omega, omega, M_0])
# convert to state vector
x_2 = convert_rv_kep.convert_rv_kep('keplerian', elems_2, 0.0, 'state').reshape(6)


# # #
# setup the simulation
# calc the orbital period
T = 2 * np.pi * np.power(a, 1.5) / np.sqrt(params['mu_E'])
dt = 10.0  # [sec]
n_orbits = 15.0
tspan = np.arange(0.0, n_orbits*T, dt)

x_0 = np.concatenate((x_1, x_2))

# solve
x_sol = integrate.odeint(func=kepler_J2_ODE.kepler_J2_ODE, t=tspan, y0=x_0, tfirst=True, args=(params, ), rtol=1.0e-12
```

```
, atol=1.0e-12)
print(x_sol.shape)
print(tspan[-1])

# plot first
font = {'family' : 'arial',
        'weight' : 'normal',
        'size'   : 12}
fig1 = plt.figure(1, figsize=(9, 4.5))
plt.rc('font', **font)
ax = fig1.gca(projection='3d')
ax.plot(x_sol[:, 0], x_sol[:, 1], x_sol[:, 2], linewidth=0.5)
ax.scatter(0.0, 0.0, 0.0, s=50, color='green')
ax.set_xlabel('X [km]')
ax.set_ylabel('Y [km]')
ax.set_zlabel('Z [km]')
#ax.set_title('First Satellite 3D Position')
plt.title('Satellite 1 Trajectory')
plt.legend(('Orbit', 'Earth'))
plt.show()


# plot second
fig2 = plt.figure(2, figsize=(9, 4.5))
ax = fig2.gca(projection='3d')
plt.rc('font', **font)
ax.plot(x_sol[:, 6], x_sol[:, 7], x_sol[:, 8], linewidth=0.5)
ax.scatter(0.0, 0.0, 0.0, s=50, color='green')
ax.set_xlabel('X [km]')
ax.set_ylabel('Y [km]')
ax.set_zlabel('Z [km]')
#ax.set_title('Second Satellite 3D Position')
plt.title('Satellite 2 Trajectory')
plt.legend(('Orbit', 'Earth'))
plt.show()

# calc & plot angular momentum and energy
momentum = np.zeros((tspan.shape[0], params['n_objs']))
energy = np.zeros((tspan.shape[0], params['n_objs']))
for i in range(tspan.shape[0]):
    for j in range(params['n_objs']):
        r = x_sol[i, 6*j:3+6*j]
        v = x_sol[i, 3+6*j:6+6*j]
        momentum[i, j] = np.linalg.norm(np.cross(r,  v))
        energy[i, j] = np.power(np.linalg.norm(v), 2.0) / 2.0 - params['mu_E'] / np.linalg.norm(r)

# print out the momentum at the right times
```

```python
n_steps = np.shape(np.arange(0.0, T, dt))[0]
for i in range(int(n_orbits)):
    for j in range(params['n_objs']):
        r_vec = x_sol[i*n_steps, 6 * j:3 + 6 * j]
        v_vec = x_sol[i*n_steps, 3 + 6 * j:6 + 6 * j]
        r = np.linalg.norm(r_vec)
        v = np.linalg.norm(v_vec)
        print(r)
        print(v)
        h_vec = np.cross(r_vec, v_vec)
        h = np.linalg.norm(h_vec)
        energy = np.power(v, 2.0) / 2.0 - params['mu_E'] / r
        #print(h)
        #print(energy)

```