```python
import numpy as np
import astropy.units as u
import astropy.constants as c
import matplotlib.pyplot as plt
from matplotlib import rc
rc('text', usetex=True)

# Load in the velocity and radius data
a = np.loadtxt('hw1_data.dat', skiprows=1)
radius = a[:, 0] * u.Unit('kpc')
vcirc = a[:, 1] * u.Unit('km/s')


### QUESTION 1 ###
# Find interior mass to a certain radius
M_int = radius * vcirc ** 2. / c.G.cgs
M_int = M_int.to('g')

# Print the answers to the first question
     ("Mass inside 1 kpc: {}".format(M_int[radius == 1.*u.kpc]/c.M_sun.cgs))
     ("Mass inside 10 kpc: {}".format(M_int[radius == 10.*u.kpc]/c.M_sun.cgs))
     ("Mass inside 20 kpc: {}".format(M_int[radius == 20.*u.kpc]/c.M_sun.cgs))


### QUESTION 2 ###
# Calculate the orbital period as a function of circular velocity and distance
T = 2. * np.pi * radius / vcirc
T = T.to('year')
     ("Orbital period at 1 kpc: {}".format(T[radius == 1.*u.kpc]))
     ("Orbital period at 10 kpc: {}".format(T[radius == 10.*u.kpc]))
     ("Orbital period at 20 kpc: {}".format(T[radius == 20.*u.kpc]))


### QUESTION 3 ###
Rh = 3 * u.Unit('kpc')  # Exponential Scale Length
# Density at the center --- ish. Fitting coefficient added
rho_0 = (M_int[0] / (np.pi * radius[0]**2.)).to(u.Unit('g/cm2')) * 0.85
# Exponential disk model for mass density at a certain radius
rho = rho_0 * np.exp(-radius / Rh)
# Enclosed mass
M_int_exp = (((-Rh * radius - Rh**2.) * np.exp(-radius / Rh) +
              Rh**2.) * 2. * np.pi * rho_0).to('g')
# Circular velocity from exponential model
vcirc_exp = (np.sqrt(c.G.cgs * M_int_exp / radius)).to('km/s')
# Circular velocity of difference between data and exponential model
M_diff = M_int - M_int_exp
vcirc_diff = (np.sqrt(c.G.cgs * M_diff / radius)).to('km/s')


# Make plots
plt.plot(radius, vcirc, 'k-', marker='o', ms=2, label='Data', alpha=0.7)
plt.plot(radius, vcirc_exp, 'r--', label='Exponential Disk Model', alpha=0.7)
plt.plot(radius, vcirc_diff, 'b--', label='Dark Matter', alpha=0.7)
plt.legend()
plt.title('Velocity Profile Contributions')
plt.xlabel('$r$ (kpc)')
plt.ylabel('$v_{c}$ (km/s)')
plt.show()
```