```
"Zhi Chen PHY540 Problem 27"

import numpy as np
import matplotlib.pyplot as plt


class problem_27:
    def __init__(self):

        # convert to m^-3
        self.rho0 = 1.e10

        # mass of proton in kg
        self.m_p = 1.67262192e-27

        # mass of electron in kg
        self.m_e = 9.1093837e-31

        # boltzmann constant in si
        self.kb = 1.380649e-23
        # self.kbeV = 8.6173e-5
        # Temperature range
        self.T = np.linspace(3000.0, 10000.0, 10000)

        self.kbT = self.kb*self.T

        # planck constant in si
        self.h = 6.62607e-34
        # self.h = 4.1357e-15

        # eV to Joule conversion
        self.eV2J = 1.60218e-19

    def K1(self):

        eps = -13.606 * self.eV2J
        lam = (self.h**2 / (2.0*np.pi*self.kbT))**(3./2.)
        mass_factor = ((self.m_p+self.m_e)/(self.m_p*self.m_e))**(3./2.)
        K1 = np.exp(-eps/self.kbT) * lam * mass_factor
        return K1

    def K2(self):

        eps = -54.417 * self.eV2J
        lam = (self.h**2 / (2.0*np.pi*self.kbT))**(3./2.)
```

```
        mass_factor = (1.0/self.m_e)**(3./2.)
        K2 = np.exp(-eps/self.kbT) * lam * mass_factor
        return K2

    def K3(self):

        eps = -1.0 * (24.587 + 54.417) * self.eV2J
        lam = (self.h**2 / (2.0*np.pi*self.kbT))**3
        mass_factor = (1.0/self.m_e**2)**(3./2.)
        K3 = np.exp(-eps/self.kbT) * lam * mass_factor
        return K3

    def rho_H(self):

        K1 = self.K1()
        rho_H = self.rho0 + 0.5*(1.0 - np.sqrt(4.0*K1*self.rho0 + 1.0))/K1
        rho_H[rho_H < 0.0] = 0.0
        return rho_H

    def rho_He(self):

        K1 = self.K1()
        K2 = self.K2()
        K3 = self.K3()
        fac = 1.0 - np.sqrt(4.0*self.rho0*K1 + 1.0)
        rho_He = self.rho0/16.0 * (fac / (fac - 2.0*K1*K2/K3))
        return rho_He

    def K4(self):
        eps = -13.606*self.eV2J
        w0 = 2321.0/0.695
        De = -2.788*self.eV2J
        r = 1.06e-10

        vib_fac = 1.0/(np.exp(w0/2.0/self.T) - np.exp(-w0/2.0/self.T))
        elec_fac = np.exp(-(De-eps)/self.kbT)

        K4 = 16.0*self.h*r*r*np.sqrt(np.pi/(self.m_p*self.kbT))*vib_fac*elec_fac
        return K4

    def K5(self):
        eps = -2.0*13.606*self.eV2J
        w0 = 4401.0/0.695
        De = -4.747*self.eV2J
        r = 0.75e-10
```

```
        vib_fac = 1.0/(np.exp(w0/2.0/self.T) - np.exp(-w0/2.0/self.T))
        elec_fac = np.exp(-(De-eps)/self.kbT)

        K5 = 8.0*self.h*r*r*np.sqrt(np.pi/(self.m_p*self.kbT))*vib_fac*elec_fac
        return K5

    def K6(self):
        eps = -(24.587+54.417)*self.eV2J
        w0 = 3161.0/0.695
        De = -2.04*self.eV2J
        r = 0.77e-10

        vib_fac = 1.0/(np.exp(w0/2.0/self.T) - np.exp(-w0/2.0/self.T))
        elec_fac = np.exp(-(De-eps)/self.kbT)

        K6 = 4.0*self.h*r*r*np.sqrt(5.0*np.pi/(2.0*self.m_p*self.kbT))*vib_fac*elec_fac
        return K6

    def rho_H2p(self):

        K4 = self.K4()
        rho_H = self.rho_H()
        rho_H2p = K4*(self.rho0 - rho_H)*rho_H
        # rho_H2p = np.log10(K4) + np.log10(self.rho0 - rho_H) + np.log10(rho_H)
        return rho_H2p

    def rho_H2(self):

        K5 = self.K5()
        rho_H = self.rho_H()
        rho_H2 = K5 * rho_H * rho_H
        # rho_H2 = np.log10(K5) + np.log10(rho_H) + np.log10(rho_H)
        return rho_H2

    def rho_HeHp(self):

        K6 = self.K6()
        rho_He = self.rho_He()
        rho_H = self.rho_H()
        rho_HeHp = K6*rho_He*(self.rho0 - rho_H)
        # rho_HeHp = np.log10(K6) + np.log10(rho_He) + np.log10(self.rho0 - rho_H)
        return rho_HeHp

    def plot(self, plot_type):

        fig = plt.figure(figsize=(16,9))
```

```
ax = fig.add_subplot(111)

if plot_type=="a":
    rhoK1 = self.rho0 * self.K1()
    rhoK2 = self.rho0 * self.K2()
    rho2K3 = self.rho0**2 * self.K3()
    ax.plot(self.T, rhoK1, "-", label=r"$\rho_0 K1$")
    ax.plot(self.T, rhoK2, "--", label=r"$\rho_0 K2$")
    ax.plot(self.T, rho2K3, "-.", label=r"$\rho_0^2 K3$")
    ax.set_yscale("log")
    ax.legend()

elif plot_type=="b":
    rho_H = self.rho_H()
    rho_He = self.rho_He()
    ax.plot(self.T, rho_H, linestyle="solid", label=r"$\rho_H$")
    ax_twin = ax.twinx()
    ax_twin.plot(self.T, rho_He, "r", linestyle="dashed", label=r"$\rho_{He}$")
    ax.set_ylabel(r"$\rho_H$ [$m^{-3}$]")
    ax_twin.set_ylabel(r"$\rho_{He}$ [$m^{-3}$]")
    ax.set_yscale("linear")
    ax_twin.set_yscale("linear")
    h1, l1 = ax.get_legend_handles_labels()
    h2, l2 = ax_twin.get_legend_handles_labels()
    ax.legend(h1+h2, l1+l2)

elif plot_type=="c":
    rhoK4 = self.rho0 * self.K4()
    rhoK5 = self.rho0 * self.K5()
    rhoK6 = self.rho0 * self.K6()
    ax.plot(self.T, rhoK4, "-", label=r"$\rho_0 K4$")
    ax.plot(self.T, rhoK5, "--", label=r"$\rho_0 K5$")
    ax.plot(self.T, rhoK6, "-.", label=r"$\rho_0 K6$")
    # ax.invert_xaxis()
    ax.set_yscale("log")
    ax.legend()

elif plot_type=="d":
    rho_H2p = self.rho_H2p()
    rho_H2 = self.rho_H2()
    rho_HeHp = self.rho_HeHp()
    ax.plot(self.T[self.T < 7500], rho_H2p[self.T < 7500],
            "-", label=r"$\rho_{H_2^{+}}$")
    ax.plot(self.T[self.T < 7500], rho_H2[self.T < 7500],
            "--", label=r"$\rho_{H_2}$")
    ax.plot(self.T, rho_HeHp, "-.", label=r"$\rho_{HeH^{+}}$")
```

```
            # ax.invert_xaxis()
            ax.set_yscale("log")
            ax.legend()

        ax.set_xlabel("T [K]")
        plt.show()

        return fig

problem27 = problem_27()
problem27.plot("a")
problem27.plot("b")
problem27.plot("c")
problem27.plot("d")
```