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

# Defining the columns of the data
data_h =(np.loadtxt("data1.txt", comments ='#',))[:, 0]
data_m =(np.loadtxt("data1.txt", comments ='#',))[:, 1]
data_s =(np.loadtxt("data1.txt", comments ='#',))[:, 2]
data_d =(np.loadtxt("data1.txt", comments ='#',))[:, 3]
data_am =(np.loadtxt("data1.txt",comments ='#',))[:, 4]
data_as =(np.loadtxt("data1.txt",comments ='#',))[:, 5]
data_mag =(np.loadtxt("data1.txt",comments ='#',))[:, 6]

# Filtering out all stars that are visible
data_hvis = data_h[np.where(data_mag <= 6)]
data_mvis = data_m[np.where(data_mag <= 6)]
data_svis = data_s[np.where(data_mag <= 6)]
data_dvis = data_d[np.where(data_mag <= 6)]
data_amvis = data_am[np.where(data_mag <= 6)]
data_asvis = data_as[np.where(data_mag <= 6)]

# Converts Right Ascension into Rad
def ra(x,y,z):
    r = np.empty(len(x))
    for i in range(len(x)):
        r[i] = (((15*np.pi)/180)*x[i]) + ((((15/60)*np.pi)/180)*y[i]) + ((((15/3600)*np.pi)
    return r

r_a = (ra(data_h, data_m, data_s))
r_avis = (ra(data_hvis, data_mvis, data_svis))

# Shifts the RAs so the span covers -pi to pi (as necessary from the Mollewiele projection)
# from the 0 to 2pi range it initially had
def r_a_prop(x):
    for i in range(len(x)):
        if x[i] > np.pi:
            x[i] = x[i] - 2*np.pi
        else:
            x[i] = x[i]
    return x

r_a_use = r_a_prop(r_a)
r_a_vuse = r_a_prop(r_avis)

# Converts Declination into Rads
def dc(x,y,z):
    r = np.empty(len(x))
    for i in range(len(x)):
        r[i] = (((np.pi)/180)*x[i]) + ((((1/60)*np.pi)/180)*y[i]) + ((((1/3600)*np.pi)/180)
    return r

d_c = (dc(data_d, data_am, data_as))
d_cvis = (dc(data_dvis, data_amvis, data_asvis))

### Filters stars that are visible and circumpolar
data_hcm = data_hvis[np.where(d_cvis > (41*(np.pi / 180)))]
data_mcm = data_mvis[np.where(d_cvis > (41*(np.pi / 180)))]
data_scm = data_svis[np.where(d_cvis > (41*(np.pi / 180)))]
```

```
data_dcm = data_dvis[np.where(d_cvis > (41*(np.pi / 180)))]
data_amcm = data_amvis[np.where(d_cvis > (41*(np.pi / 180)))]
data_ascm = data_asvis[np.where(d_cvis > (41*(np.pi / 180)))]

r_a_cm = (ra(data_hcm, data_mcm, data_scm))
r_a_cmuse = r_a_prop(r_a_cm)
d_ccm = (dc(data_dcm, data_amcm, data_ascm))

#Plain scatter plot of all data
plt.scatter(r_a, d_c, c='r', s=0.1)
plt.title("Respective Right Ascension and Declination of stars seen in the night sky", font
plt.xlabel("Rigth Ascension of stars (in Rads)", fontsize=16)
plt.ylabel("Declination of stars (in Rads)", fontsize=16)
plt.show()

# Molleweilde projection of stars
fig = plt.figure(num=1, figsize=(10, 5))
ax = fig.add_subplot(111, projection="mollweide")
plt.title("Molleweide projection of stars", fontsize=20)
ax.scatter(r_a_use, d_c, s=0.01, marker=".", color="red", label="All Stars")
ax.scatter(r_a_vuse, d_cvis, s=0.2, marker="*", color="blue", label="Visible Stars")
ax.scatter(r_a_cmuse, d_ccm, s=0.5, marker="s", color="black", label="Circumpolar Stars")
plt.xlabel("Right Ascension (in Rads)", fontsize=16)
plt.ylabel("Declination (in Rads)", fontsize=16)
plt.legend(loc="best", fontsize="large", markerscale=12)
plt.show()

print("The number of Visible stars that can be seen is {}".format(len(r_a_vuse)))
print("The number of circumpolar stars that can be seen from Vancouver is {}".format(len(r_
```