```
#!/usr/bin/python
import sys, math, cmath
import numpy

Numbers2Atoms ={
1:'H', # iupac '97 in water
2:'He',
3:'Li',
4:'Be',
5:'B',
6:'C',
7:'N',
8:'O',
9:'F',
10:'Ne',
11:'Na',
12:'Mg',
13:'Al',
14:'Si',
15:'P',
16:'S',
17:'Cl',
18:'Ar',
19:'K',
20:'Ca',
21:'Sc',
22:'Ti',
23:'V',
24:'Cr',
25:'Mn',
26:'Fe',
27:'Ni',
28:'Co',
29:'Cu',
30:'Zn',
31:'Ga',
32:'Ge',
33:'As',
34:'Se',
35:'Br',
36:'Kr',
37:'Rb',
38:'Sr',
39:'Y',
40:'Zr',
41:'Nb',
```

42:'Mo',
43:'Tc',
44:'Ru',
45:'Rh',
46:'Pd',
47:'Ag',
48:'Cd',
49:'In',
50:'Sn',
51:'Sb',
52:'Te',
53:'I',
54:'Xe',
55:'Cs',
56:'Ba',
57:'La',
58:'Ce',
59:'Pr',
60:'Nd',
61:'Pm',
62:'Sm',
63:'Eu',
64:'Gd',
65:'Tb',
66:'Dy',
67:'Ho',
68:'Er',
69:'Tm',
70:'Yb',
71:'Lu',
72:'Hf',
73:'Ta',
74:'W',
75:'Re',
76:'Os',
77:'Ir',
78:'Pt',
79:'Au',
80:'Hg',
81:'Tl',
82:'Pb',
83:'Bi',
84:'Po',
85:'At',
86:'Rn',
87:'Fr',

```
88:'Ra',
89:'Ac',
90:'Th',
91:'Pa',
92:'U',
93:'Np',
94:'Pu',
95:'Am',
96:'Cm',
97:'Bk',
98:'Cf',
99:'Es',
100:'Fm',
101:'Md',
102:'No',
103:'Lr',
104:'Rf',
105:'Db',
106:'Sg'
}


PeriodicTable ={
'H':[1,1,[1.0078250321,2.0141017780],[0.999885,0.0001157]], # iupac '97 in water
'He':[2,0,[3.0160293097,4.0026032497],[0.00000137,0.99999863]], # iupac iso '97
'Li':[3,1,[6.0151233,7.0160040],[0.0759,0.9241]], # iupac '97
'Be':[4,2,[9.0121821],[1.0]], # iupac '97
'B':[5,3,[10.0129370,11.0093055],[0.199,0.801]], # iupac' 97
'C':[6,-4,[12.0,13.0033548378],[0.9893,0.0107]], # iupac '97
'N':[7,5,[14.0030740052,15.0001088984],[0.99632,0.00368]], # iupac '97
'O':[8,-2,[15.9949146221,16.99913150,17.9991604],[0.99757,0.00038,0.00205]], #
iupac '97
'F':[9,-1,[18.99840320],[1.0]], # iupac '97
'Ne':[10,0,[19.9924401759,20.99384674,21.99138551],[0.9048,0.0027,0.0925]], #
iupac '97 in air
'Na':[11,1,[22.98976967],[1.0]], #iupac '97
'Mg':[12,2,[23.98504190,24.98583702,25.98259304],[0.7899,0.10,0.1101]], #iupac
'97
'Al':[13,3,[26.98153844],[1.0]], #iupac '97
'Si':[14,4,[27.9769265327,28.97649472,29.97377022],[0.92297,0.046832,0.03087
2]],#iupac '97
'P':[15,5,[30.97376151],[1.0]], #iupac '97
'S':[16,-
2,[31.97207069,32.97145850,33.96786683,35.96708088],[0.9493,0.0076,0.0429,0.
0002]], #iupac '97
'Cl':[17,-1,[34.96885271,36.96590260],[0.7578,0.2422]], #iupac '97
```

'Ar':[18,0,[35.96754628,37.9627322,39.962383123],[0.003365,0.000632,0.996003]],#iupac '97 in air
'K':[19,1,[38.9637069,39.96399867,40.96182597],[0.932581,0.000117,0.067302]], #iupac '97
'Ca':[20,2,[39.9625912,41.9586183,42.9587668,43.9554811,45.9536928,47.952534],[0.96941,0.00647,0.00135,0.02086,0.00004,0.00187]], #iupac '97
'Sc':[21,3,[44.9559102],[1.0]], #iupac '97
'Ti':[22,4,[45.9526295,46.9517638,47.9479471,48.9478708,49.9447921],[0.0825,0.0744,0.7372,0.0541,0.0518]], #iupac '97
'V':[23,5,[49.9471628,50.9439637],[0.00250,0.99750]], #iupac '97
'Cr':[24,2,[49.9460496,51.9405119,52.9406538,53.9388849],[0.04345,0.83789,0.09501,0.02365]], #iupac '97
'Mn':[25,2,[54.9380496],[1.0]], #iupac '97
'Fe':[26,3,[53.9396148,55.9349421,56.9353987,57.9332805],[0.05845,0.91754,0.02119,0.00282]], #iupac '97
'Ni':[27,2,[57.9353479,59.9307906,60.9310604,61.9283488,63.9279696],[0.680769,0.262231,0.011399,0.036345,0.009256]], #iupac '97
'Co':[28,2,[58.933195],[1.0]], #iupac '97
'Cu':[29,2,[62.9296011,64.9277937],[0.6917,0.3083]], #iupac '97
'Zn':[30,2,[63.9291466,65.9260368,66.9271309,67.9248476,69.925325],[0.4863,0.2790,0.0410,0.1875,0.0062]], #iupac '97
'Ga':[31,3,[68.925581,70.9247050],[0.60108,0.39892]], #iupac '97
'Ge':[32,2,[69.9242504,71.9220762,72.9234594,73.9211782,75.9214027],[0.2084,0.2754,0.0773,0.3628,0.0761]], #iupac '97
'As':[33,3,[74.9215964],[1.0]], #iupac '97
'Se':[34,4,[73.9224766,75.9192141,76.9199146,77.9173095,79.9165218,81.916700],[0.0089,0.0937,0.0763,0.2377,0.4961,0.0873]], #iupac '97
'Br':[35,-1,[78.9183376,80.916291],[0.5069,0.4931]],#iupac '97
'Kr':[36,0,[77.920386,79.916378,81.9134846,82.914136,83.911507,85.9106103],[0.0035,0.0228,0.1158,0.1149,0.5700,0.1730]], #iupac '97 in air
'Rb':[37,1,[84.9117893,86.9091835],[0.7217,0.2783]], #iupac '97
'Sr':[38,2,[83.913425,85.9092624,86.9088793,87.9056143],[0.0056,0.0986,0.0700,0.8258]], #iupac '97
'Y': [39,3,[88.9058479],[1.0]], #iupac '97
'Zr':
[40,4,[89.9047037,90.9056450,91.9050401,93.9063158,95.908276],[0.5145,0.1122,0.1715,0.1738,0.0280]],#iupac '97
'Nb':[41,5,[92.9063775],[1.0]], #iupac '97
'Mo':[42,6,[91.906810,93.9050876,94.9058415,95.9046789,96.9060210,97.9054078,99.907477],[0.1484,0.0925,0.1592,0.1668,0.0955,0.2413,0.0963]], #checked, iupac '97
'Tc': [43,2,[96.906365,97.907216,98.9062546],[1.0]], #no natural abundance
'Ru':
[44,3,[95.907598,97.905287,98.9059393,99.9042197,100.9055822,101.9043495,103.905430],[0.0554,0.0187,0.1276,0.1260,0.1706,0.3155,0.1862]], #iupac '97
'Rh':[45,2,[102.905504],[1.0]], #iupac '97

'Pd':[46,2,[101.905608,103.904035,104.905084,105.903483,107.903894,109.905152],[0.0102,0.1114,0.2233,0.2733,0.2646,0.1172]], #iupac '97
'Ag':[47,1,[106.905093,108.904756],[0.51839,0.48161]], #iupac '97
'Cd':[48,2,[105.906458,107.904183,109.903006,110.904182,111.9027572,112.9044009,113.9033581,115.904755],[0.0125,0.0089,0.1249,0.1280,0.2413,0.1222,0.2873,0.0749]],#iupac '97
'In':[49,3,[112.904061,114.903878],[0.0429,0.9571]], #iupac '97
'Sn':[50,4,[111.904821,113.902782,114.903346,115.901744,116.902954,117.901606,118.903309,119.9021966,121.9034401,123.9052746],[0.0097,0.0066,0.0034,0.1454,0.0768,0.2422,0.0859,0.3258,0.0463,0.0579]], #iupac '97
'Sb':[51,3,[120.9038180,122.9042157],[0.5721,0.4279]], #iupac '97
'Te':[52,4,[119.904020,121.9030471,122.9042730,123.9028195,124.9044247,125.9033055,127.9044614,129.9062228],[0.0009,0.0255,0.0089,0.0474,0.0707,0.1884,0.3174,0.3408]],#iupac '97
'I':[53,-1,[126.904468],[1.0]], #iupac '97
'Xe':[54,0,[123.9058958,125.904269,127.9035304,128.9047795,129.9035079,130.9050819,131.9041545,133.9053945,135.907220],[0.0009,0.0009,0.0192,0.2644,0.0408,0.2118,0.2689,0.1044,0.0887]], #iupac '97
'Cs':[55,1,[132.905447],[1.0]], #iupac '97
'Ba':[56,2,[129.906310,131.905056,133.904503,134.905683,135.904570,136.905821,137.905241],[0.00106,0.00101,0.02417,0.06592,0.07854,0.11232,0.71698]], #iupac '97
'La':[57,3,[137.907107,138.906348],[0.00090,0.99910]],#iupac '97
'Ce':[58,3,[135.907140,137.905986,139.905434,141.909240],[0.00185,0.00251,0.88450,0.11114]],#iupac '97
'Pr':[59,3,[140.907648],[1.0]], #iupac '97
'Nd':[60,3,[141.907719,142.909810,143.910083,144.912569,145.913112,147.916889,149.920887],[0.272,0.122,0.238,0.083,0.172,0.057,0.056]],#iupac '97
'Pm':[61,3,[144.91270],[1.0]], #no natural occurence
'Sm':[62,3,[143.911995,146.914893,147.914818,148.917180,149.917271,151.919728,153.922205],[0.0307,0.1499,0.1124,0.1382,0.0738,0.2675,0.2275]], #iupac '97
'Eu':[63,3,[150.919846,152.921226],[0.4781,0.5219]], #iupac '97
'Gd':[64,3,[151.919788,153.920862,154.922619,155.922120,156.923957,157.924101,159.927051],[0.0020,0.0218,0.1480,0.2047,0.1565,0.2484,0.2186]],#iupac '97
'Tb':[65,4,[158.925343],[1.0]], #iupac '97
'Dy':[66,3,[155.924278,157.924405,159.925194,160.926930,161.926795,162.928728,163.929171],[0.0006,0.0010,0.0234,0.1891,0.2551,0.2490,0.2818]], #iupac '97
'Ho':[67,3,[164.930319],[1.0]], #iupac '97
'Er':[68,3,[161.928775,163.929197,165.930290,166.932045,167.932368,169.935460],[0.0014,0.0161,0.3361,0.2293,0.2678,0.1493]], #iupac '97
'Tm':[69,3,[168.934211],[1.0]], #iupac '97
'Yb':[70,3,[167.933894,169.934759,170.936322,171.9363777,172.9382068,173.9388581,175.942568],[0.0013,0.0304,0.1428,0.2183,0.1613,0.3183,0.1276]], #iupac '97
'Lu':[71,3,[174.9407679,175.9426824],[0.9741,0.0259]],#iupac '97

```
'Hf':[72,4,[173.940040,175.9414018,176.9432200,177.9436977,178.9458151,179.
9465488],[0.0016,0.0526,0.1860,0.2728,0.1362,0.3508]], #iupac '97
'Ta':[73,5,[179.947466,180.947996],[0.00012,0.99988]], #iupac '97
'W':[74,6,[179.946704,181.9482042,182.9502230,183.9509312,185.9543641],[0.0
012,0.2650,0.1431,0.3064,0.2843]], #iupac  '97
'Re':[75,2,[184.9529557,186.9557508],[0.3740,0.6260]],#iupac '97
'Os':[76,4,[183.952491,185.953838,186.9557479,187.9558360,188.9581449,189.9
58445,191.961479],[0.0002,0.0159,0.0196,0.1324,0.1615,0.2626,0.4078]],#iupac
'97
'Ir':[77,4,[190.960591,192.962924],[0.373,0.627]], #iupac '97
'Pt':[78,4,[189.959930,191.961035,193.962664,194.964774,195.964935,197.9678
76],[0.00014,0.00782,0.32967,0.33832,0.25242,0.07163]],#iupac '97
'Au':[79,3,[196.966552],[1.0]], #iupac '97
'Hg':[80,2,[195.965815,197.966752,198.968262,199.968309,200.970285,201.9706
26,203.973476],[0.0015,0.0997,0.1687,0.2310,0.1318,0.2986,0.0687]], #iupac '97
'Tl':[81,1,[202.972329,204.974412],[0.29524,0.70476]], #iupac '97
'Pb':[82,2,[203.973029,205.974449,206.975881,207.976636],[0.014,0.241,0.221,0.
524]],#
'Bi':[83,3,[208.980383],[1.0]], #iupac '97
'Po':[84,4,[209.0],[1.0]],
'At':[85,7,[210.0],[1.0]],
'Rn':[86,0,[220.0],[1.0]],
'Fr':[87,1,[223.0],[1.0]],
'Ra':[88,2,[226.0],[1.0]],
'Ac':[89,3,[227.0],[1.0]],
'Th':[90,4,[232.0380504],[1.0]], #iupac '97
'Pa':[91,4,[231.03588],[1.0]],
'U':[92,6,[234.0409456,235.0439231,236.0455619,238.0507826],[0.000055,0.0072
00,0.0,0.992745]], #iupac '97
'Np':[93,5,[237.0],[1.0]],
'Pu':[94,3,[244.0],[1.0]],
'Am':[95,2,[243.0],[1.0]],
'Cm':[96,3,[247.0],[1.0]],
'Bk':[97,3,[247.0],[1.0]],
'Cf':[98,0,[251.0],[1.0]],
'Es':[99,0,[252,.0],[1.0]],
'Fm':[100,0,[257.0],[1.0]],
'Md':[101,0,[258.0],[1.0]],
'No':[102,0,[259.0],[1.0]],
'Lr':[103, 0,[262.0],[1.0]],
'Rf':[104, 0,[261.0],[1.0]],
'Db':[105, 0,[262.0],[1.0]],
'Sg':[106, 0,[266.0],[1.0]]
}

def xyzgetfck(infile):
```

```
items=0
countlines=0
atoms=[]
xyz=[]
f=open(infile,'r')
found=False
for line in f:
		line=line.rstrip('\n')
		if found==True:
			data=filter(None,line.split(' '))
			for n in range(0,len(data)):

				atoms+=[Numbers2Atoms[int(data[n])]]

				countlines+=1

		if "Atomic numbers" in line:
			found=True
			data=filter(None,line.split(' '))
			items=int(data[data.index("N=")+1])

		if countlines==items:
			found=False
f.seek(0)

countlines=0
found=False
for line in f:
		line=line.rstrip('\n')
		if found==True:
			data=filter(None,line.split(' '))
			for n in range(0,len(data)):

				xyz+=[float(data[n])*0.52918]
				countlines+=1

		if "Current cartesian coordinates" in line:
			found=True
			data=filter(None,line.split(' '))
			items=int(data[data.index("N=")+1])

		if countlines==items:
			found=False


f.close()
```

```
        cartesian=[]
        for n in range(0,len(atoms)):
                x_str='%10.5f'% xyz[3*n]
                y_str='%10.5f'% xyz[3*n+1]
                z_str='%10.5f'% xyz[3*n+2]
                element=str(atoms[n])
                cartesian+=[element+(3-len(element))*' '+10*' '+\
                (8-len(x_str))*' '+x_str+10*' '+(8-len(y_str))*' '+y_str+10*' '+(8-
len(z_str))*' '+z_str+'\n']

        return [cartesian]


def xyzgetrawdata(infile):
        f=open(infile,'r')

        n=0
        preamble=[]

        struct=[]
        for line in f:
                if n<2:
                        preamble+=[line.rstrip()]
                if n>1:
                        line=line.rstrip()
                        struct+=[line]
                n+=1
        xyz=[struct]

        return xyz, preamble

def xyzgetelements(rawdata):

        n=0
        for structure in rawdata:
                n=n+1
                elements=[]

                for item in structure:
                        coordx=filter(None,item.split(' '))
                        coordy=filter(None,item.split('\t'))
                        if len(coordx)>len(coordy):
                                coords=coordx
                        else:
                                coords=coordy
                        elements+=[coords[0]]
```

```
        return elements

def xyzgetIsotope(myatom):
        isotope_ratio=PeriodicTable[myatom][3]
        isotope_masses=PeriodicTable[myatom][2]
        mass = isotope_masses[isotope_ratio.index(max(isotope_ratio))]
        return mass


def xyzgetmasses(elements):
    massfile2=""
    for i in range(0,len(elements)):
                myelement=xyzgetIsotope(elements[i])
                massfile2+=str("%1.7E" % myelement).replace('E','D')+'\n'
        return massfile2

def xyzgetmasseso18(elements):
        massfile1=""
        for i in range(0,len(elements)):
                if elements[i]!='O':
                        myelement=xyzgetIsotope(elements[i])
                else:
                        myelement=17.9991604
                massfile1+=str("%1.7E" % myelement).replace('E','D')+'\n'
        return massfile1



def fckgetrawdata(infile):
        items=0
        countlines=0
        flathessian=[]
        f=open(infile,'r')
        found=False
        for line in f:
                        line=line.rstrip('\n')
                        if found==True:
                                data=filter(None,line.split(' '))
                                for n in range(0,len(data)):
                                        flathessian+=[float(data[n])]
                                        countlines+=1

                        if "Cartesian Force Constants" in line:
                                found=True
                                data=filter(None,line.split(' '))
                                items=int(data[data.index("N=")+1])
```

```
                        if countlines==items:
                                found=False

        f.close()
        return flathessian

def fckmakeflathessian(rawdata):
        hessfile=""
        for n in range(0,len(rawdata)):
                        formdata="%1.10e" % (rawdata[n])
                        if float(formdata)>0.0:
                                formdata=' '+formdata
                formdata=formdata.replace('e','D')
                        hessfile+='    '+str(formdata)+'\n'
        return hessfile

def hesscalcfreq(massweighted):
        cau=(2.99792458 * 2.418884326505)*1E-9 #c in metres per t(a.u.)
        amu2au=(1.66053892/9.10938291)*1E4 # 1 amu in a.u. (via kgs)
        cmtom=1.0/100.0 #m per cm
        scaling=cmtom*(1/(2*math.pi*cau*math.sqrt(amu2au))) #( m/cm *
1/((m/au) * au) = m/cm * 1/m = 1/cm)
        eigenval,eigenvec=numpy.linalg.eig(massweighted);
        freqs=[scaling*cmath.sqrt(x) for x in sorted(eigenval)]

        return freqs

def hessreadmasses(massfile):

        newmass=massfile.split('\n')
        masses=[]
        natoms=len(massfile)

        for n in newmass:

                n=n.replace('D','E')
                if n.rstrip('\n')!='':
                        masses+=[float(n.rstrip('\n'))]

        return masses

def hessreadhess(hessfile, masses):
        newhess=hessfile.split('\n')
        natoms=len(masses)
```

```
        line=-1
        hessian=numpy.zeros((natoms*3,natoms*3))
        massweighted=numpy.zeros((natoms*3, natoms*3))

        for k in range(0,natoms*3):
                for l in range(0,k+1):
                        line+=1

                        n=newhess[line]
                        n=n.replace('D','E')

                        if n.rstrip('\n')!='':
                                hessian[l,k]=float(n.rstrip('\n'))
                                massweighted[l,k]=hessian[l,k]/math.sqrt(
masses[int(math.ceil(l/3))] * masses[int(math.ceil(k/3))] ) ;


        line=-1

        for k in range(0,natoms*3):
                for l in range(0,k+1):
                        line+=1
                        n=newhess[line]

                        n=n.replace('D','E')
                        if n.rstrip('\n')!='':
                                hessian[k,l]=float(n.rstrip('\n'))
                                massweighted[k,l]=massweighted[l,k];


        return massweighted

def hessmassweight(hessian,masses):
        weighted=0

        return weighted

def rpfrreaddata(infile):
        f=infile.split('\n')
        data=[]
        for n in f:
                lu=n.rstrip('\n.')
                lu=lu.split()
                if len(lu)==2:
                        data+=[[float(lu[0]), float(lu[1])]]
        return data
```

```
def rpfrcalcrpfr(data,temperature):
        #define variables, constants
        #gas constant
        r=8.314459848;
        #speed of light
        c=29979245800.0;
        #avogadro's constant
        a=6.02214085774E23;
        #planck's constant
        h=6.62607004081E-34;

        hc=h*c*a
        rt=r*temperature

        prod=1.0
        summ=0.0
        rtsum2=0.0

        outfile=""
        for n in range(0,len(data)):
                u=hc*data[n][0]/(rt)
                up=hc*data[n][1]/(rt)
          if (u and up) != 0:
             numerator=(u*math.exp(-u/2.0)/(1.0-math.exp(-u)))
                denominator=(up*math.exp(-up/2.0)/(1.0-math.exp(-up)))
            prod=prod*numerator/denominator
                    rtsum2=rtsum2+math.log(up/u)

                    summ=summ+math.log(numerator/denominator)
                    print "%5d \t %12.4f \t %12.4f \t %20.10E \t %12.6f \t %12.6f" %
(n+1, data[n][0], data[n][1], (numerator/denominator)-1.0, prod, summ)
                    outfile+="%5d \t %12.4f \t %12.4f \t %20.10E \t %12.6f \t
%12.6f\n" % (n+1, data[n][0], data[n][1], (numerator/denominator)-1.0, prod,
summ)

        return rtsum2, outfile

if __name__ == "__main__":

        fckinfile=sys.argv[1]
        temperature=float(sys.argv[2])
#First xyz2masses
        xyz=xyzgetfck(fckinfile)

        #Save cartesian structure
```

```
        f=open('in.xyz','w')
        f.write(str(len(xyz[0]))+'\n'+'\n')
        for n in xyz[0]:
                f.write(n)
        f.close()

        elements=xyzgetelements(xyz)

        massfile2=xyzgetmasses(elements)
        massfile1=xyzgetmasseso18(elements)

        #Save mass files
        f=open('light.mass','w')
        f.write(str(len(massfile2.split('\n'))-1)+'\n')
        for n in massfile2:
                f.write(n)
        f.close()
        f=open('heavy.mass','w')
        f.write(str(len(massfile1.split('\n'))-1)+'\n')
        for n in massfile1:
                f.write(n)
        f.close()

#Then fck in

        rawdata=fckgetrawdata(fckinfile)
        hessfile=fckmakeflathessian(rawdata)

        #save flat hessian
        f=open('in.hess','w')
        f.write(hessfile)
        f.close()

#Then hess + mass -> freq

        masses1=hessreadmasses(massfile1)
        masses2=hessreadmasses(massfile2)
        massweighted1=hessreadhess(hessfile, masses1)
        massweighted2=hessreadhess(hessfile, masses2)

        freqs=hesscalcfreq(massweighted1)
        freqs2=hesscalcfreq(massweighted2)

        rpfrinfile=""

        for n in range(0,len(freqs)):
```

```
                rpfrinfile+='%.15e\t%.15e\n' % (freqs[n].real-freqs[n].imag,
freqs2[n].real-freqs2[n].imag)

        #write frequencies
        f=open('out.freq','w')
        f.write(str(len(freqs))+'\n')
        f.write(rpfrinfile)
        f.close()


#Then freq + temp -> rpfr

        data=rpfrreaddata(rpfrinfile)
        rpfr,outfile=rpfrcalcrpfr(data,temperature)

        f=open('rpfr.out','w')
        f.write(outfile)
        f.write(str(rpfr)+'\n')
        f.close()

        print rpfr
```