```
#!/usr/bin/python
import numpy as np
import pylab

def dft(y):
    N = len(y)
    c = np.zeros(N//2+1,complex)
    for k in range(N//2+1):
        for n in range(N):
            c[k]+=y[n]*np.exp(-2j*np.pi*k*n/N)
    return c

def part_a():
    data = np.loadtxt('sunspots.txt')
    print("approx: {}".format(133.2))
    pylab.plot(data[:,0], data[:,1])
    pylab.show()

def part_b():
    data = np.loadtxt('sunspots.txt')
    coef = np.fft.rfft(data[:,1])
    pylab.plot(np.power(np.abs(coef), 2))
    pylab.show()

def part_c():
    data = np.loadtxt('sunspots.txt')
    coef = list(np.fft.rfft(data[:,1]))
    disp = [np.abs(x)**2 for x in coef]
    peak = disp.index(max(disp[2:]))

def main():
    part_a()
    part_b()
    part_c()

if __name__ == "__main__":
    main()
```

```
 1: #!/usr/bin/python
 2: import numpy as np
 3: import pylab
 4:
 5: def part_a():
 6:     piano = np.loadtxt("piano.txt")
 7:     trumpet = np.loadtxt("trumpet.txt")
 8:     fft_piano = np.fft.rfft(piano)
 9:     fft_trumpet = np.fft.rfft(trumpet)
10:     pylab.plot(fft_piano[:10000])
11:     pylab.plot(fft_trumpet[:10000])
12:     pylab.show()
13:
14: def part_b():
15:     pass
16:
17: def main():
18:     part_a()
19:     part_b()
20:
21: if __name__ == "__main__":
22:     main()
```

```
 1: #!/usr/bin/python
 2: import numpy as np
 3: import pylab
 4:
 5: def part_a():
 6:     data = np.loadtxt('dow.txt')
 7:     pylab.plot(data)
 8:     pylab.show()
 9:
10: def part_b():
11:     data = np.loadtxt('dow.txt')
12:     fft = np.fft.rfft(data)
13:     pylab.plot(fft)
14:     pylab.show()
15:
16: def part_c():
17:     data = np.loadtxt('dow.txt')
18:     fft = np.fft.rfft(data)
19:     fft[int(len(fft)*0.1):] = [0] * (len(fft) - int(len(fft)*0.1))
20:     pylab.plot(fft)
21:     pylab.show()
22:
23: def part_d():
24:     data = np.loadtxt('dow.txt')
25:     fft = np.fft.rfft(data)
26:     fft[int(len(fft)*0.1):] = [0] * (len(fft) - int(len(fft)*0.1))
27:     new_data = np.fft.irfft(fft)
28:     pylab.plot(data)
29:     pylab.plot(new_data)
30:     pylab.show()
31:
32: def part_e():
33:     data = np.loadtxt('dow.txt')
34:     fft = np.fft.rfft(data)
35:     fft[int(len(fft)*0.02):] = [0] * (len(fft) - int(len(fft)*0.02))
36:     new_data = np.fft.irfft(fft)
37:     pylab.plot(data)
38:     pylab.plot(new_data)
39:     pylab.show()
40:
41: def main():
42:     part_a()
43:     part_b()
44:     part_c()
45:     part_d()
46:     part_e()
47:
48: if __name__ == "__main__":
49:     main()
```

```
 1: #!/usr/bin/python
 2: import numpy as np
 3: import pylab
 4:
 5:
 6: def gen_data():
 7:     return [
 8:         1 if np.floor(2 * x) % 2 == 0 else -1 for x in np.linspace(0, 1, 1000)
 9:     ]
10:
11: def main():
12:     data = gen_data()
13:     fft = np.fft.rfft(data)
14:     fft[10:] = [0] * (len(fft) - 10)
15:     new_data = np.fft.irfft(fft)
16:     pylab.plot(data)
17:     pylab.plot(new_data)
18:     pylab.show()
19:
20: if __name__ == "__main__":
21:     main()
22:
```

```
 1: #!/usr/bin/python
 2: import numpy as np
 3: import pylab
 4:
 5: def dct(y):
 6:     N = len(y)
 7:     y2 = np.empty(2*N,float)
 8:     y2[:N] = y[:]
 9:     y2[N:] = y[::-1]
10:
11:     c = np.fft.rfft(y2)
12:     phi = np.exp(-1j*np.pi*np.arange(N)/(2*N))
13:     return np.real(phi*c[:N])
14:
15: def idct(a):
16:     N = len(a)
17:     c = np.empty(N+1,complex)
18:
19:     phi = np.exp(1j*np.pi*np.arange(N)/(2*N))
20:     c[:N] = phi*a
21:     c[N] = 0.0
22:     return np.fft.irfft(c)[:N]
23:
24: def part_a():
25:     data = np.loadtxt('dow2.txt')
26:     fft = np.fft.rfft(data)
27:     fft[int(len(fft)*0.02):] = [0] * (len(fft) - int(len(fft)*0.02))
28:     new_data = np.fft.irfft(fft)
29:     pylab.plot(data)
30:     pylab.plot(new_data)
31:     pylab.show()
32:
33: def part_b():
34:     data = np.loadtxt('dow2.txt')
35:     fft = dct(data)
36:     fft[int(len(fft)*0.02):] = [0] * (len(fft) - int(len(fft)*0.02))
37:     new_data = idct(fft)
38:     pylab.plot(data)
39:     pylab.plot(new_data)
40:     pylab.show()
41:
42: def main():
43:     part_a()
44:     part_b()
45:
46: if __name__ == "__main__":
47:     main()
```

```
#!/usr/bin/python
import numpy as np
import pylab


def gaussian(x, y, Lx, Ly):
    return sum([
        sum([
            np.exp(-((x + (n * Lx))**2 + (y + (m * Ly))**2) / (2 * (25**2)))
            for n in range(-1, 1)
        ])
        for m in range(-1, 1)
    ])

def part_a():
    img_data = np.loadtxt('blur.txt')
    pylab.imshow(img_data)
    pylab.set_cmap('Greys_r')
    pylab.show()


def part_b():
    img_data = np.loadtxt('blur.txt')
    shape = img_data.shape
    blur = [[gaussian(x, y, shape[0], shape[1])
             for x in range(shape[0])]
            for y in range(shape[1])]
    pylab.imshow(blur)
    pylab.set_cmap('Greys_r')
    pylab.show()


def part_c():
    img_data = np.loadtxt('blur.txt')
    shape = img_data.shape
    blur = [[gaussian(x, y, shape[0], shape[1])
             for x in range(shape[0])]
            for y in range(shape[1])]
    fft_img = np.fft.rfft2(img_data)
    fft_blur = np.fft.rfft2(blur)
    fft_unblured = fft_img
    for i, row in enumerate(fft_unblured):
        for j, elem in enumerate(row):
            if fft_blur[i][j] > 1e-3:
                fft_unblured[i][j] /= (fft_blur[i][j])
    unblured = np.fft.irfft2(fft_unblured)
    pylab.imshow(unblured)
    pylab.set_cmap('Greys_r')
    pylab.show()


def main():
    part_a()
    part_b()
    part_c()


if __name__ == "__main__":
    main()
```