In [1]:
```
from PIL import Image
import numpy as np
import os
from matplotlib.pyplot import imshow

def read_data(dir_path):
    x = np.array([])
    file_names = os.listdir(dir_path)
    elem_shape = []
    num_elem = 0
    for fname in file_names:
        num_elem += 1
        img = np.asarray(Image.open(dir_path + '/' + fname).convert('L'))
        img_shape = np.shape(img)
        img = np.reshape(img, [img_shape[0]*img_shape[1], 1])
        elem_shape = img_shape
        if x.size == 0:
            x = img
        else:
            x = np.concatenate([x, img], axis=1)
    return (x, elem_shape, num_elem)
```

In [2]:
```
dir_path = './problem_statement/dataset'

values, e_sz, num = read_data(dir_path)
```

In [3]:
```
def PCA(X, K=-1):
    if K == -1:
        K = np.shape(X)[1]
    sz = np.shape(X)
    M = np.mean(X, axis=0)
    CX = X - M
    COV = CX.T.dot(CX)
    eigen_values, eigen_vecs = np.linalg.eig(COV)
    pseudo_eigen_vecs = CX.dot(eigen_vecs)
    data = []
    for idx, val in enumerate(eigen_values):
        data.append((np.array(val), np.reshape(pseudo_eigen_vecs[:, idx], [sz[0
], 1])))
    data.sort(key=lambda pair: pair[0], reverse=True)
    ei_vals = np.array([])
    ei_vecs = np.array([])
    for val in data:
        if ei_vals.size == 0:
            ei_vals = np.array([val[0]])
            ei_vecs = np.array(val[1])
        else:
            ei_vals = np.concatenate([ei_vals, np.array([val[0]])])
            ei_vecs = np.concatenate([ei_vecs, val[1]], axis=1)
    return (ei_vals[0:K], ei_vecs[:, 0:K])
```

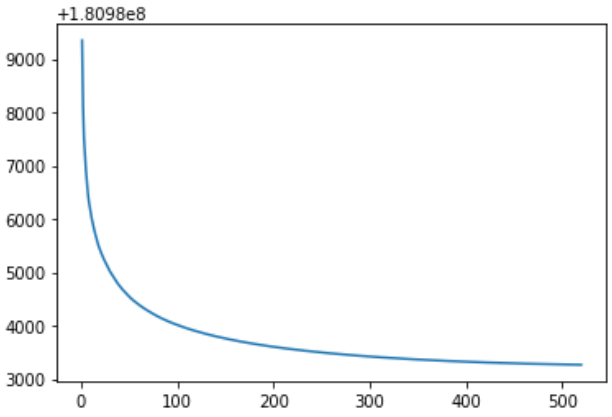
In [4]:
```
vals, vecs = PCA(values)
```

In [8]:
```
T = 77
mean_sq_err = []
num_ei_vecs = []
save_img = []
clustering_eg = []
counter = 1
proj_values = np.zeros(np.shape(values))
M = np.mean(values, axis=0)
C_values = values - M
for v in vecs.T:
    abs_v = np.sum(np.sqrt(np.square(v)))
    v = v / abs_v
    alpha = np.array([v]).dot(C_values)
    if len(clustering_eg) < 3:
        clustering_eg.append(alpha)
    proj_values += (alpha.T.dot(np.array([v]))).T
    save_img.append(proj_values[:, T] + M[T])
    err = (C_values - proj_values)*(C_values - proj_values) / np.shape(vals)[0]
    err = np.sum(np.sum(err))
    mean_sq_err.append(err)
    num_ei_vecs.append(counter)
    counter += 1
```

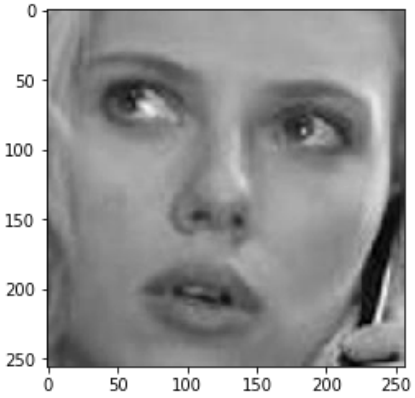
In [6]:
```
import matplotlib.pyplot as plt

print(np.shape(mean_sq_err))
plt.plot(num_ei_vecs, mean_sq_err)
```

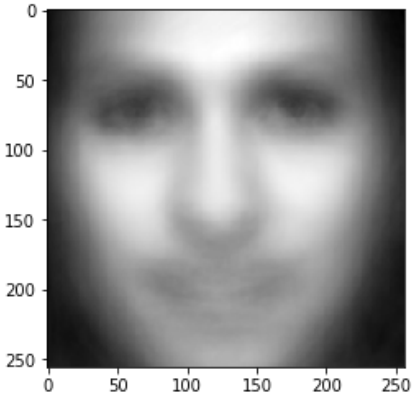
```
(520,)
```

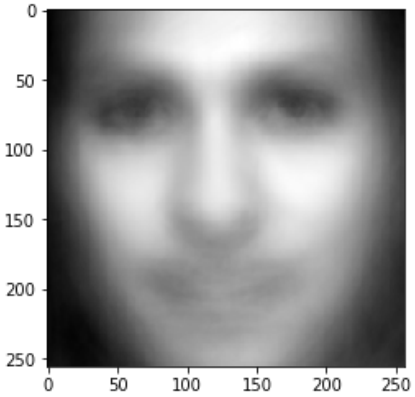
Out[6]:
```
[<matplotlib.lines.Line2D at 0x7f1969c514a8>]
```

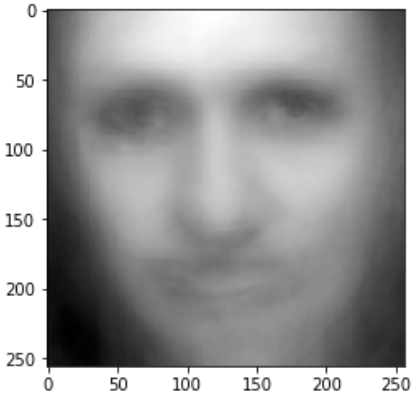


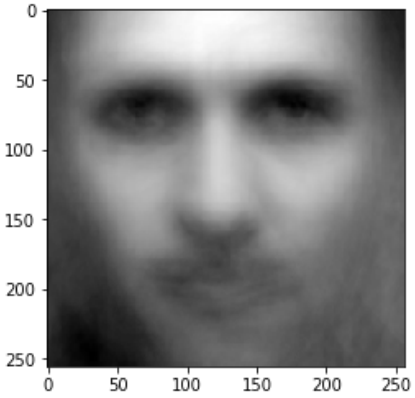
In [7]:
```
from matplotlib.pyplot import imsave

def reconstruct(M, shape):
    return np.reshape(M, shape)

orig_img = reconstruct(values[:, T], e_sz)
plt.figure()
imshow(orig_img, cmap='gray')
imsave('./pca_images/original.png', orig_img, cmap='gray')

for idx, im in enumerate(save_img[0:15]):
    orig_img = reconstruct(im + M[T], e_sz)
    plt.figure()
    imshow(orig_img, cmap='gray')


for idx, im in enumerate(save_img):
    orig_img = reconstruct(im + M[T], e_sz)
    imsave('./pca_images/' + str(idx+1) + '.png', orig_img, cmap='gray')
```

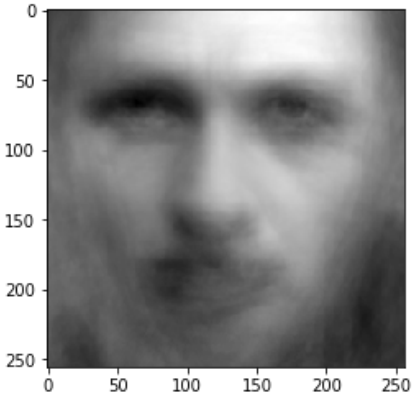

0
50
100
150
200
250
0
50
100
150
200
250

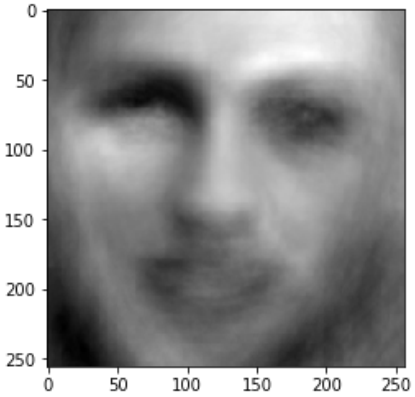

0
50
100
150
200
250
0
50
100
150
200
250

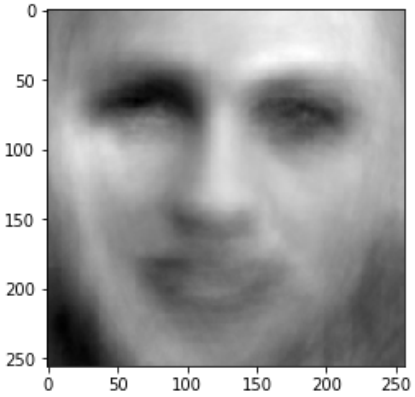

0
50
100
150
200
250
0
50
100
150
200
250

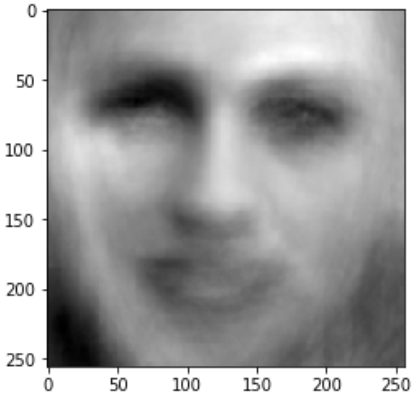

0
50
100
150
200
250
0
50
100
150
200
250

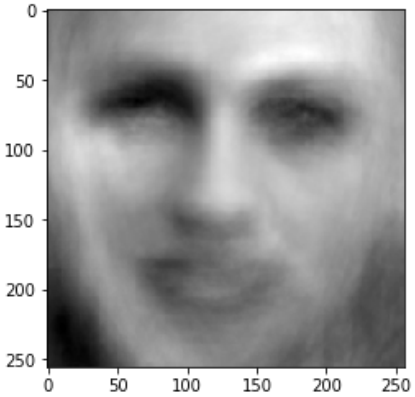

0
50
100
150
200
250
0
50
100
150
200
250

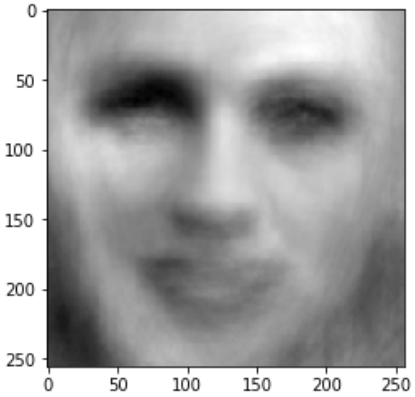

0
50
100
150
200
250
0
50
100
150
200
250

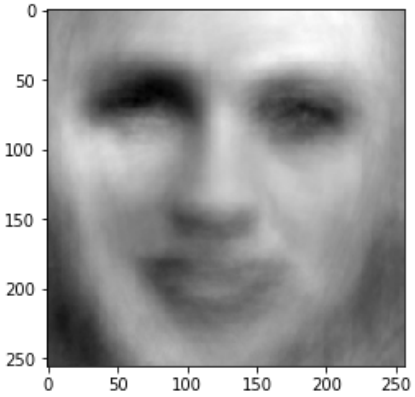

0
50
100
150
200
250
0
50
100
150
200
250

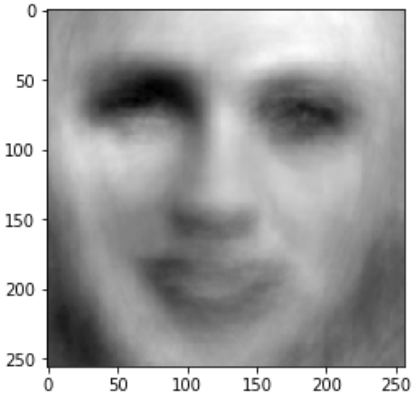

0
50
100
150
200
250
0
50
100
150
200
250

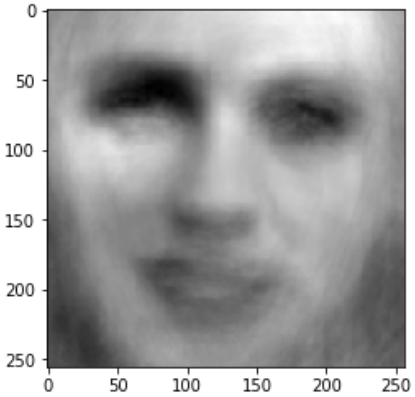

0
50
100
150
200
250
0
50
100
150
200
250

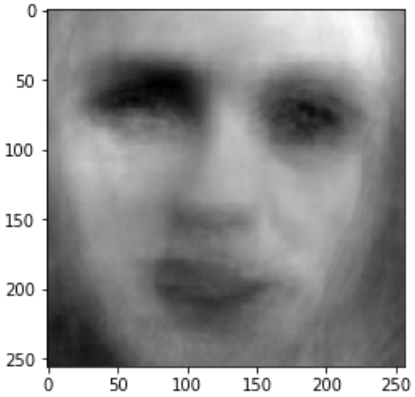

0
50
100
150
200
250
0
50
100
150
200
250

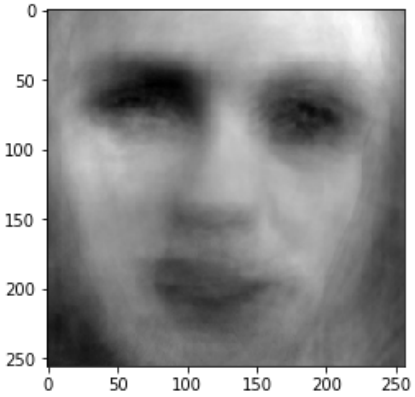

0
50
100
150
200
250
0
50
100
150
200
250


0
50
100
150
200
250
0
50
100
150
200
250


0
50
100
150
200
250
0
50
100
150
200
250


0
50
100
150
200
250
0
50
100
150
200
250


0
50
100
150
200
250
0
50
100
150
200
250


0
50
100
150
200
250
0
50
100
150
200
250

In [30]:
```
import matplotlib
from mpl_toolkits.mplot3d import Axes3D

plt.figure()
plt.scatter(clustering_eg[0][0], np.zeros_like(clustering_eg[0][0]))
plt.figure()
plt.scatter(clustering_eg[0][0], clustering_eg[1][0])

fig = plt.figure()
ax = plt.axes(projection='3d')
ax.scatter3D(clustering_eg[0][0], clustering_eg[1][0], clustering_eg[2][0], c=c
lustering_eg[2][0])
```

Out[30]: <mpl_toolkits.mplot3d.art3d.Path3DCollection at 0x7f1969a21358>