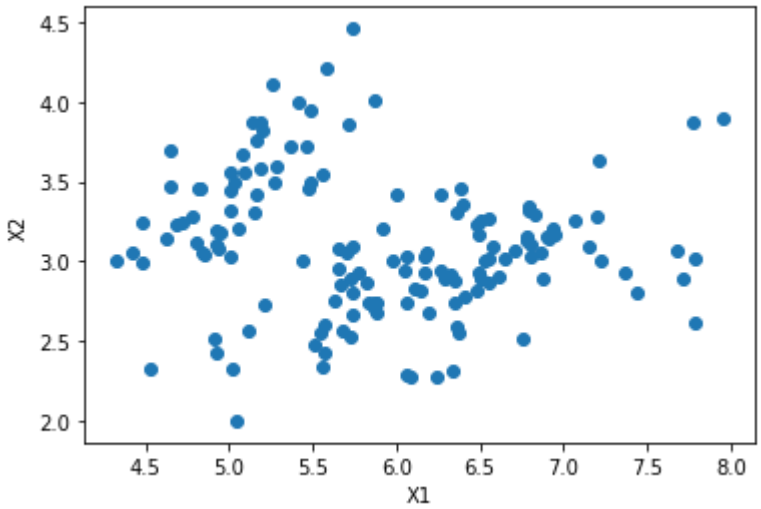
# Problem 1

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import mltools as ml
```

## Problem 1.1

In [2]:
```
iris = np.genfromtxt("data/iris.txt",delimiter=None)
```

In [3]:
```
X1 = iris[:,0]
X2 = iris[:,1]
plt.scatter(X1,X2)
plt.xlabel('X1')
plt.ylabel('X2')
plt.show()
```



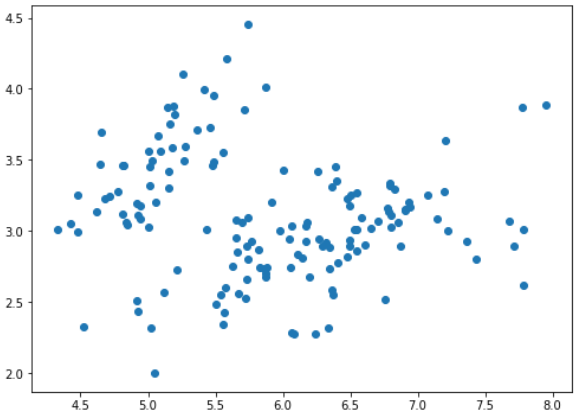

I think there are two clusters----upper left and bottom right.

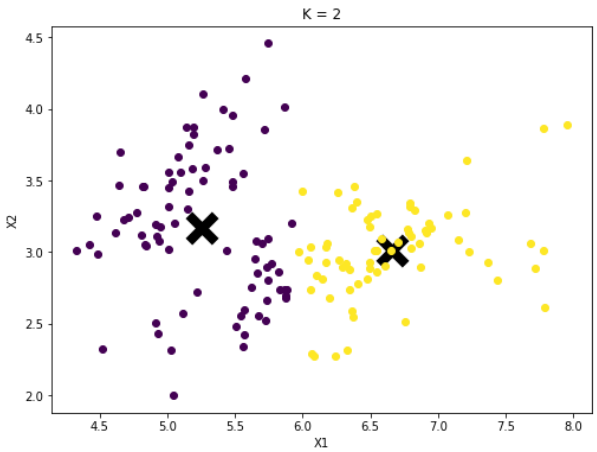
Because there the divide line between them is obvious which is a slope.

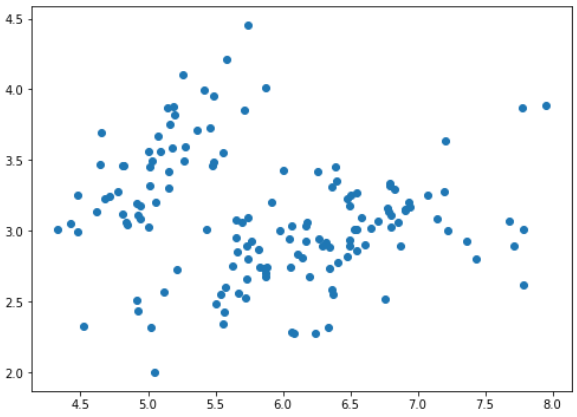
## Problem 1.2

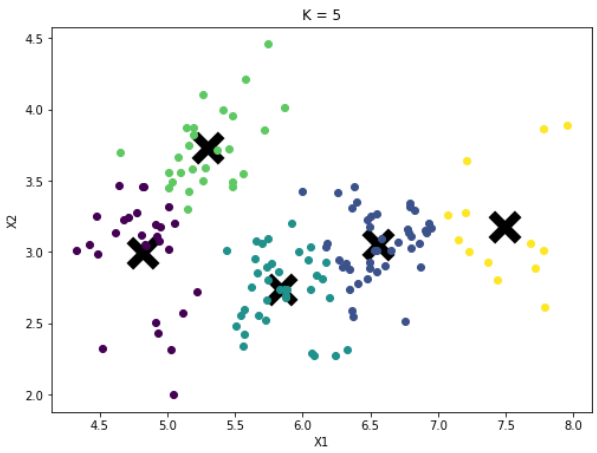
In [4]:
```
import sys
n_clusters = [2,5,20]
s = sys.maxsize
```

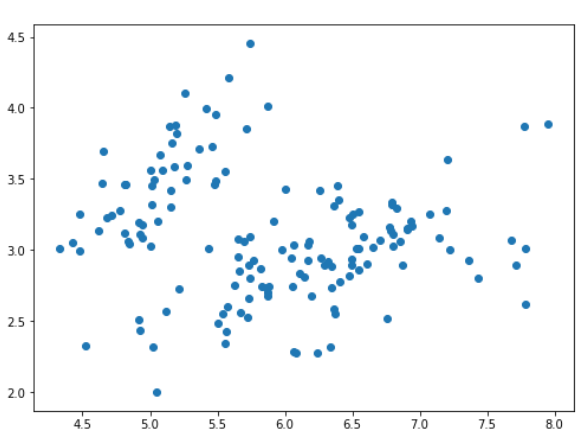
In [5]:
```
for i in n_clusters:
    for j in range(5):
        Z1, mu1, ssd1 = ml.cluster.kmeans(iris[:,:2], K=i)
        if ssd1 < s:
            s = ssd1
        Z = Z1
        mu = mu1
    fig, ax = plt.subplots(1, 2, figsize=(18, 6))
    ml.plotClassify2D(None, iris[:,:2], Z)
    plt.title('K = '+ str(i))
    plt.xlabel('X1')
    plt.ylabel('X2')
    ax[0].scatter(X1, X2)
    ax[1].scatter(mu[:, 0], mu[:, 1], s=500, marker='x', facecolor='black', lw=8) # Plotti
```

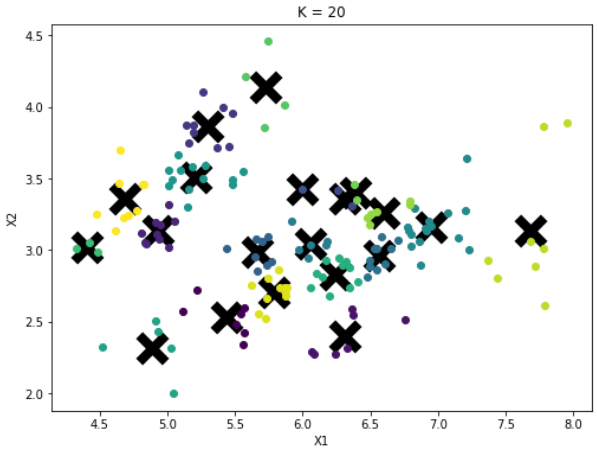


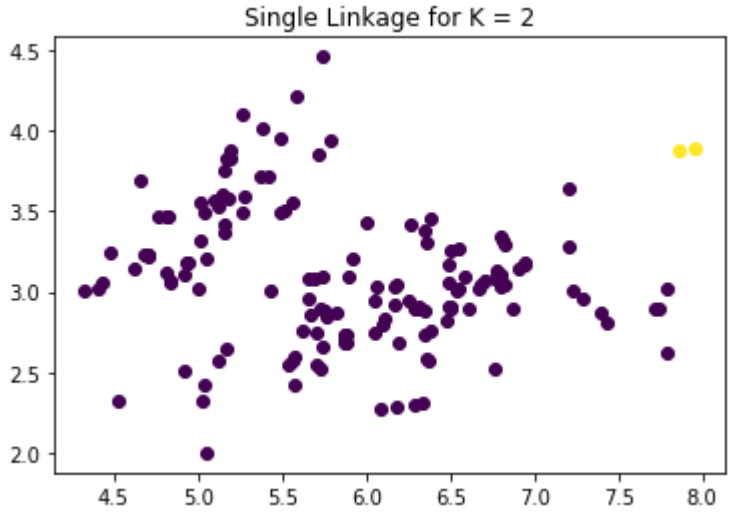



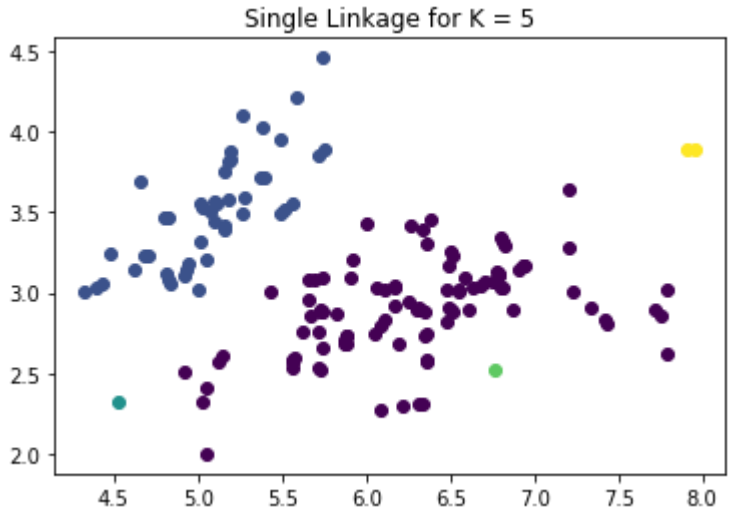

4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0

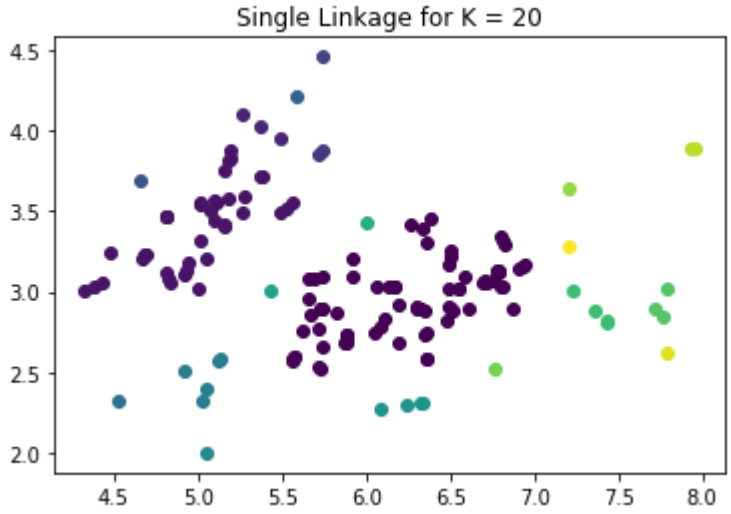

K = 20
X2
X1
4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0

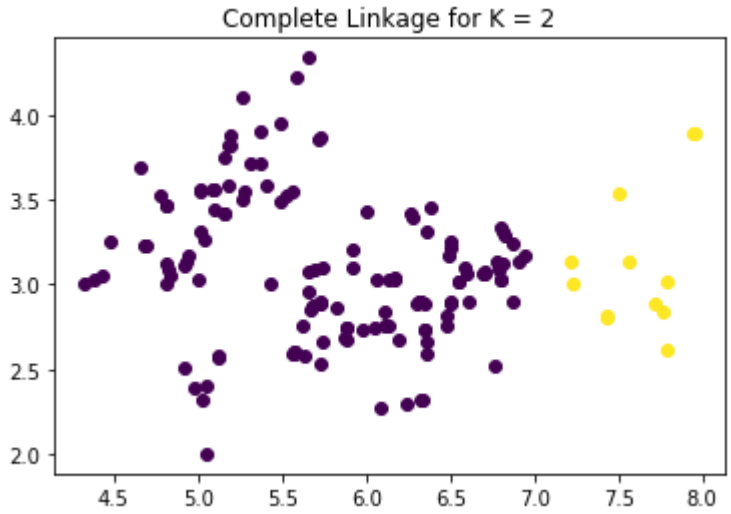
## Problem 1.3

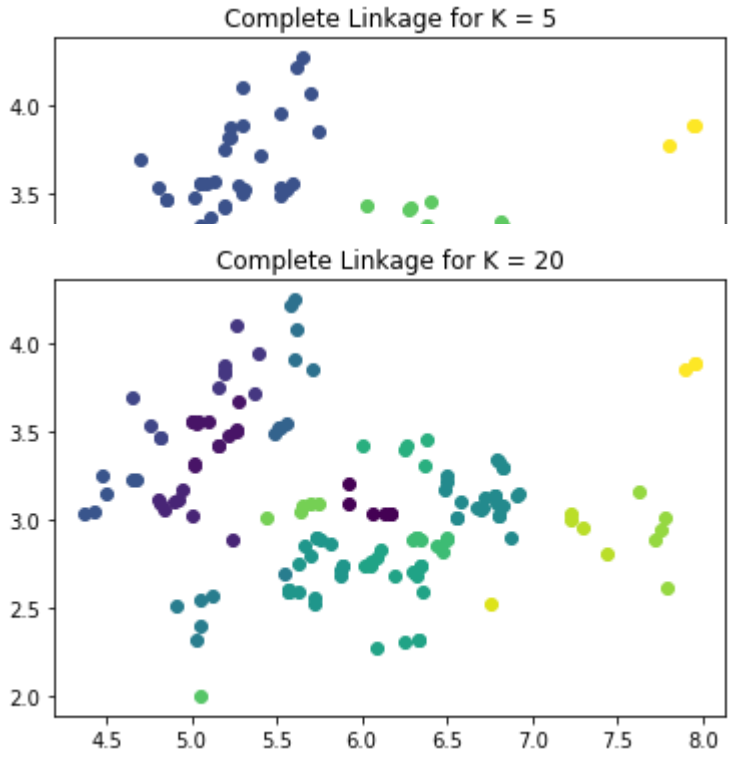
In [6]:
```
z, j = ml.cluster.agglomerative(iris[:,:2], 2, method='min')
plt.title("Single Linkage for K = 2");
ml.plotClassify2D(None, iris[:,:2], z);
plt.show()

z, j = ml.cluster.agglomerative(iris[:,:2], 5, method='min')
plt.title("Single Linkage for K = 5");
ml.plotClassify2D(None, iris[:,:2], z);
plt.show()

z, j = ml.cluster.agglomerative(iris[:,:2], 20, method='min')
plt.title("Single Linkage for K = 20");
ml.plotClassify2D(None, iris[:,:2], z);
plt.show()

z, j = ml.cluster.agglomerative(iris[:,:2], 2, method='max')
plt.title("Complete Linkage for K = 2");
ml.plotClassify2D(None, iris[:,:2], z);
plt.show()

z, j = ml.cluster.agglomerative(iris[:,:2], 5, method='max')
plt.title("Complete Linkage for K = 5");
ml.plotClassify2D(None, iris[:,:2], z);
plt.show()

z, j = ml.cluster.agglomerative(iris[:,:2], 20, method='max')
plt.title("Complete Linkage for K = 20");
ml.plotClassify2D(None, iris[:,:2], z);
plt.show()
```




Single Linkage for K = 5
4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0


Single Linkage for K = 20
4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0


Complete Linkage for K = 2
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0




### Problem 1.4

Similarities: Both categorize data according to distance somewhat.

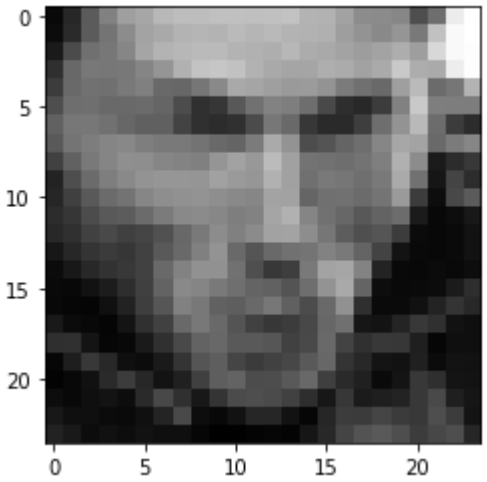
Differences: It seems like k-means algorithm did better than agglomerative clustering

## Problem 2

### Problem 2.1

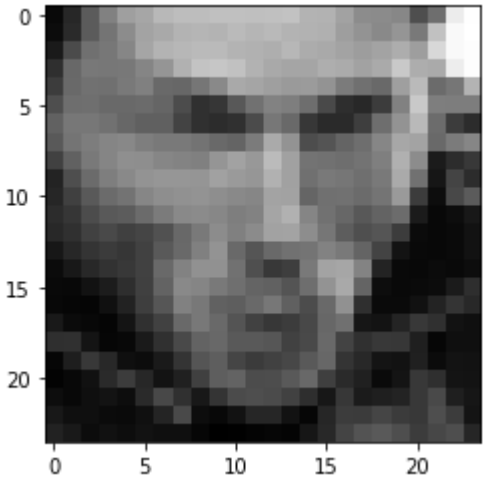
In [7]:
```
X = np.genfromtxt("data/faces.txt", delimiter=None) # load face dataset
plt.figure()
img = np.reshape(X[i,:],(24,24)) # convert vectorized data to 24x24 image patches
plt.imshow( img.T , cmap="gray")
```

Out[7]: <matplotlib.image.AxesImage at 0x19580b80108>




In [8]:
```
mean = X.mean()
X = X - mean
plt.figure()
img = np.reshape(X[i,:],(24,24)) # convert vectorized data to 24x24 image patches
plt.imshow( img.T , cmap="gray")
```

Out[8]: <matplotlib.image.AxesImage at 0x19580ae9e88>



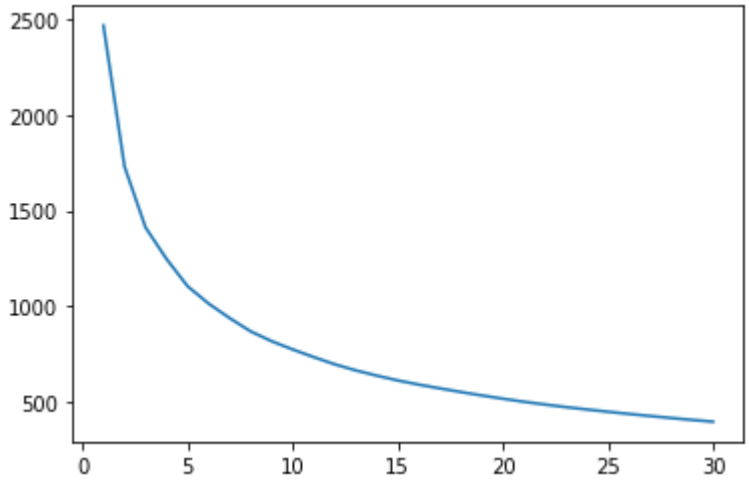

## Problem 2.2

In [9]:
```
import scipy.linalg
U, S, V = scipy.linalg.svd(X, full_matrices=False)
W = U.dot(np.diag(S))
print(W.shape)
print(V.shape)
```

```
(4916, 576)
(576, 576)
```

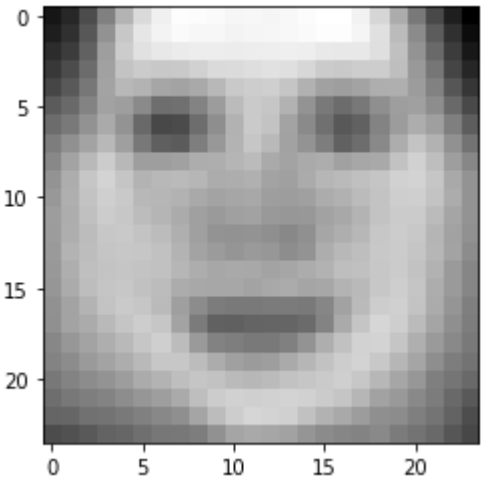
## Problem 2.3

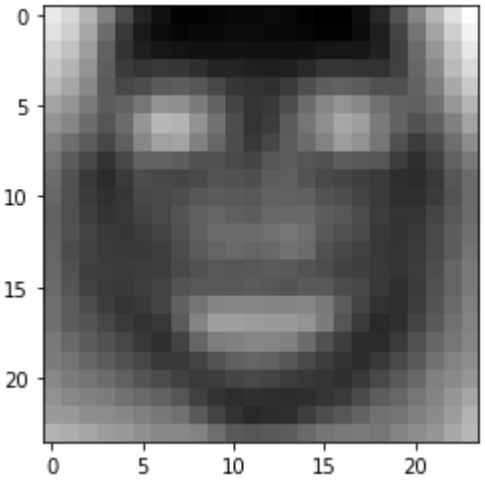
In [10]:
```
mse = []
for k in range(1, 31):
    X0 = W[:, :k].dot(V[:k,:])
    mse.append(np.mean((X - X0)**2))
fig, ax = plt.subplots()
ax.plot(range(1,31), mse)
plt.show()
```

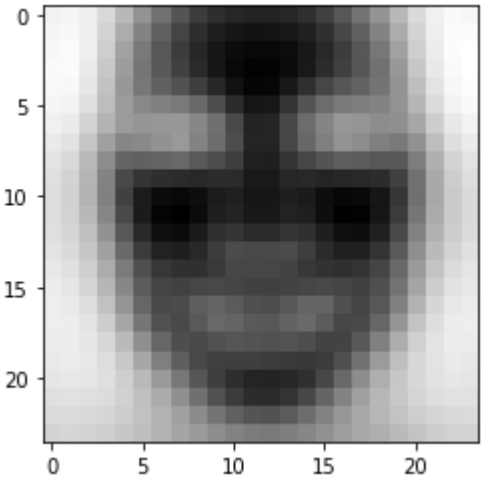


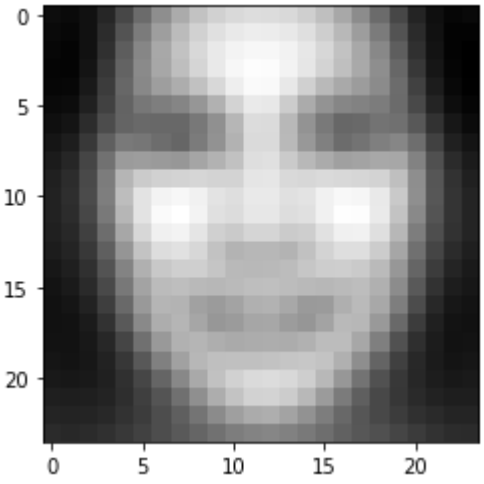

## Problem 2.4

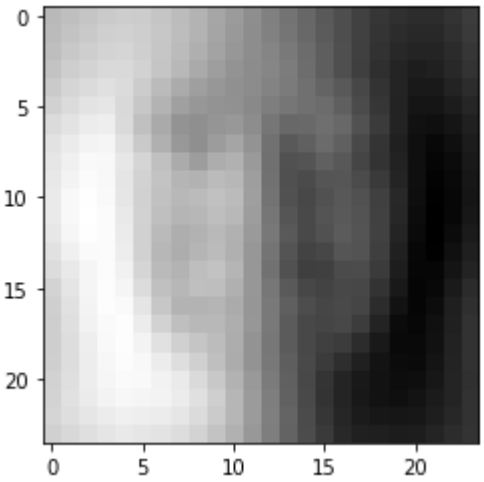
In [11]:
```
for i in range(3):
    sf = 2*np.median(np.abs(W[:,i]))
    img1 = np.reshape(mean + sf*V[i,:],(24,24))
    img2 = np.reshape(mean - sf*V[i,:],(24,24))
    plt.imshow( img1.T , cmap="gray")
    plt.show()
    plt.imshow( img2.T , cmap="gray")
    plt.show()
```

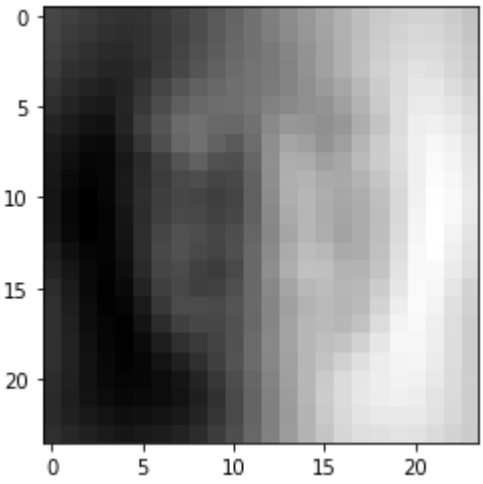




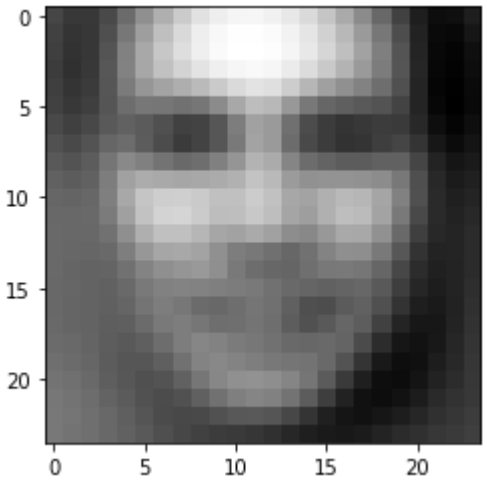



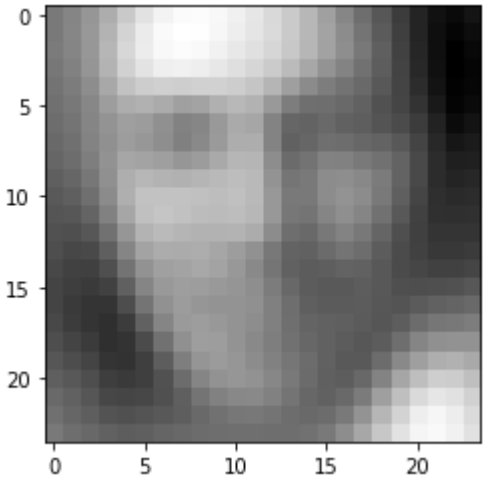

## Problem 2.5

In [12]:
```
print("First image")
for k in [5, 10, 50, 100]:
    print("K = "+str(k))
    X0 = np.dot(W[0,:k], V[:k,:])
    img = np.reshape(X0,(24,24)) # convert vectorized data point t
    plt.imshow( img.T , cmap="gray")
    plt.show()
```

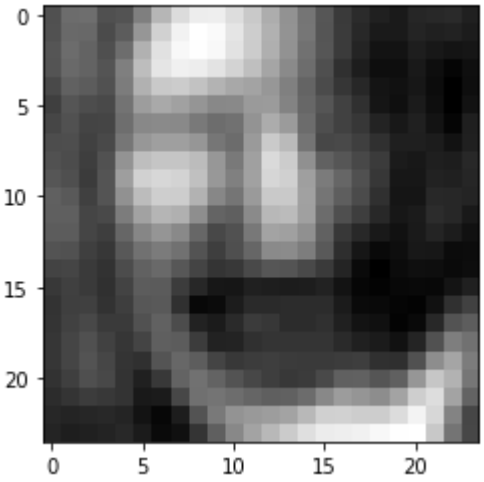
```
First image
K = 5
```



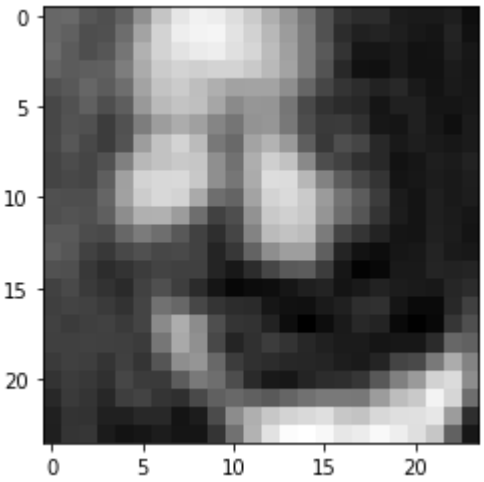

```
K = 10
```




```
K = 50
```

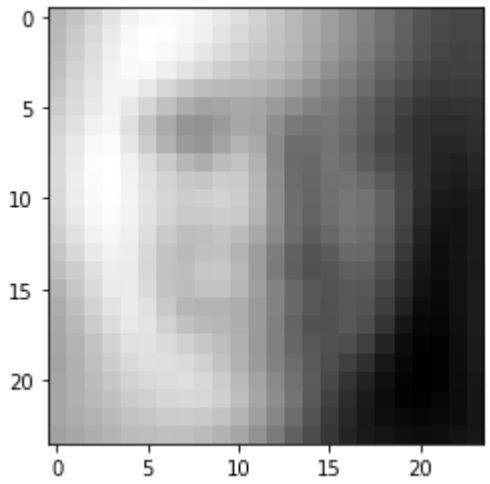



K = 100

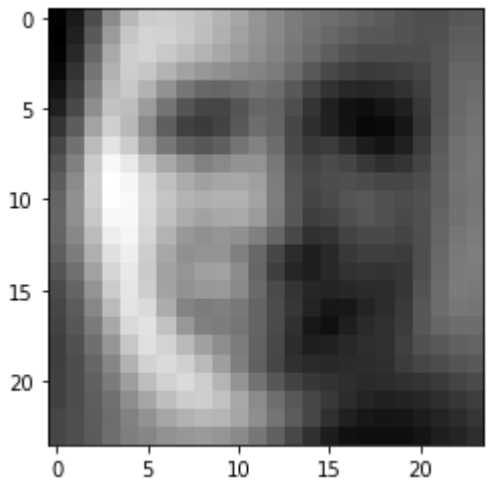


In [13]:
```
print("Second image")
for k in [5, 10, 50, 100]:
    print("K = "+str(k))
    X0 = np.dot(W[1,:k], V[:k,:])
    img = np.reshape(X0,(24,24)) # convert vectorized data point t
    plt.imshow( img.T , cmap="gray")
    plt.show()
```

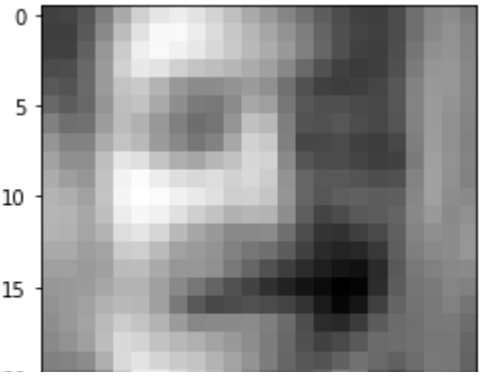
```
Second image
K = 5
```



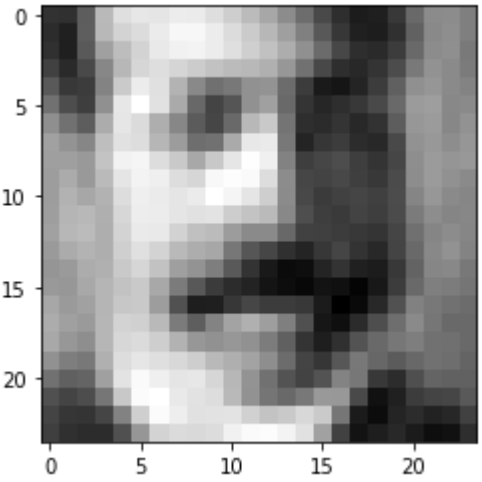

```
K = 10
```




```
K = 50
```

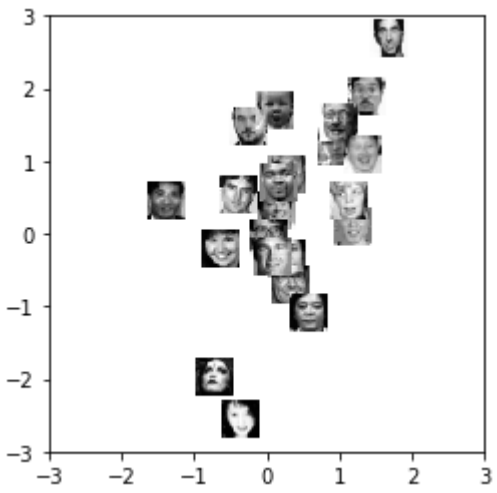



K = 100




## Problem 2.6

In [14]:
```
from numpy import random
idx = random.randint(len(iris), size=(25))

import mltools.transforms
coord,params = ml.transforms.rescale( W[:,0:2] ) # normalize scale of "W" locations
plt.figure();
for i in idx:
# compute where to place image (scaled W values) & size
    loc = (coord[i,0],coord[i,0]+0.5, coord[i,1],coord[i,1]+0.5)
    img = np.reshape( X[i,:], (24,24) ) # reshape to square
    plt.imshow( img.T , cmap="gray", extent=loc ) # draw each image
    plt.axis( (-3,3,-3,3) ) # se
```




# Problem 3

I have followed the academic honesty guidelines posted on the course website

In [ ]:
```

```