In [1]:
```
from skimage.io import imread
import matplotlib.pyplot as plt

img = imread('C:/Users/burcu/Burcu Havza Yağlı.jpeg');
plt.imshow(img)
plt.show()
```

8 bit bir resim içinde tespit etmek istediğimiz bir detay seçin ve o bölgeyi korelasyon filtresi ile tespit edin.

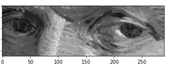
Not: gerçek resimler ile çalışırken, korelasyon öncesinde f ve h matrislerini bir ortalama alıcı filtreden geçirip, çıktıları orjinal matrislerden çıkararak matris ortalamalarını sıfıra çekebilirsiniz.

Bu işlem sonrasında korelasyon alırsanız yanlış korelasyon tepelerinden kurtulabilirsiniz. Derste yazılan olasılık teorisindeki korelasyon formülünde hatırlarsanız matrisler ortalamadan çıkarılıyordu.

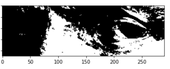
not1: Ortalama çıkarmaya örnek: mask=ones(3,3)/9; f = f - filter2(mask, f); Dikkat edin burada yerel ortalama alınmıştır. Bu soruda korelasyon matrisinde doğru noktada yüksek bir tepe değer görmelisiniz...

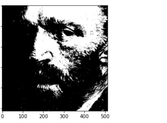
not2: matris sınırlarında yüksek korelasyonlar elde edebilirsiniz. Onlar ihmal edilebilir.

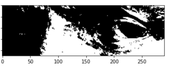
In [22]:
```
from skimage.color import rgb2gray
from skimage.io import imread
import matplotlib.pyplot as plt
import numpy as np
import cv2

image = cv2.imread('C:/Users/burcu/van_gogh.jpg', 0);

f=image
plt.imshow(f, cmap='gray')
plt.show()

#h=image[125:180, 105:170]
h=image[90:180, 90:180]
plt.imshow(h, cmap='gray')
plt.show()
```

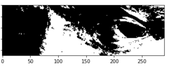


In [23]:
```
import cv2
import numpy as np
from skimage import data
import matplotlib.pyplot as plt
from skimage.color import rgb2gray
from skimage.io import imread
import array as arr

image[image>=127]=255
image[image<127]=0

#h=image[125:195, 105:170]
h=image[90:180, 90:380]
h=h-np.mean(h)
h_pad=np.pad(h, ((0, image.shape[0]-h.shape[0]),(0, image.shape[1]-h.shape[1])))

plt.imshow(h, cmap='gray')
plt.show()

f=f-np.mean(f)
plt.imshow(f, cmap='gray')
plt.show()

f_fft=np.fft.fft2(image)
h_fft=np.conjugate(np.fft.fft2(h_pad))
G=(f_fft*h_fft)
res_g=np.fft.ifft2(G)

sorted_list=sorted(np.round(abs(np.real(res_g).ravel())), reverse=True)
coordinate_list=list()

for row in range(res_g.shape[0]):
    for col in range(res_g.shape[1]):
        #a=np.round(abs(np.real(res_g[row][col])))
        #print(a)
        if np.round(abs(np.real(res_g[row][col]))) == sorted_list[0] or np.round(np.abs(np.real(res_g[row][col]))) == sorted_list[1]:
            coordinate_list.append([row, col])

for coordinate in coordinate_list:
    result=image[coordinate[0]:coordinate[0]+h.shape[0], coordinate[1]:coordinate[1]+h.shape[1]]
    plt.figure()
    plt.imshow(result, cmap='gray')
```









-->Burada Van Gogh'un gözlerini alıp h=h-np.mean(h) yaptığımızda çıktının daha da netleştiğini gözlemleyebiliriz. Burada çok net bir şekilde anlaşılmasa da aşağıda "a" harfini yazı içerisinde ararken h=h-np.mean(h) yaptığımızda netliği daha rahat gözlemleyebiliriz.

-->h_pad işleminde h'a yani filtremize zero padding işlemi uygulanır.

-->Daha sonra hem h', i hem de f'i yani image'i fast fourier transformuna tabi tutarız ve FFT'deki çıktılar üzerinde nokta çarpımı uygularız. Ancak burada gözden kaçırmamamız gereken şey h'ın konjugesini kullanmamız gerektiğidir. Konjuge aldığımızda correlation işemi olur. Burada konjuge almasaydık convolusyon işlemi uygulamış olurduk.

-->Aşağıda yazı içerisinde birden fazla aynı harfi bulabiliriz.

In [20]:
```
from skimage.color import rgb2gray
from skimage.io import imread
import matplotlib.pyplot as plt
import numpy as np
import cv2

image = cv2.imread('C:/Users/burcu/hbr.png', 0);

f=image
plt.imshow(f, cmap='gray')
plt.show()

h=image[125:195, 105:170]
plt.imshow(h, cmap='gray')
plt.show()

image[image>=127]=255
image[image<127]=0

h=image[125:195, 105:170]
h=h-np.mean(h)
h_pad=np.pad(h, ((0, image.shape[0]-h.shape[0]),(0, image.shape[1]-h.shape[1])))

plt.imshow(h, cmap='gray')
plt.show()

f=f-np.mean(f)
plt.imshow(f, cmap='gray')
plt.show()

f_fft=np.fft.fft2(image)
h_fft=np.conjugate(np.fft.fft2(h_pad))
G=(f_fft*h_fft)
res_g=np.fft.ifft2(G)

sorted_list=sorted(np.round(abs(np.real(res_g).ravel())), reverse=True)
coordinate_list=list()


for row in range(res_g.shape[0]):
    for col in range(res_g.shape[1]):
        #a=np.round(abs(np.real(res_g[row][col])))
        #print(a)
        if np.round(abs(np.real(res_g[row][col]))) == sorted_list[0] or np.round(np.abs(np.real(res_g[row][col]))) == sorted_list[1]:
            coordinate_list.append([row, col])

for coordinate in coordinate_list:
    result=image[coordinate[0]:coordinate[0]+h.shape[0], coordinate[1]:coordinate[1]+h.shape[1]]
    plt.figure()
    plt.imshow(result, cmap='gray')
```