Problem 2

In [119]:

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
from skimage.feature import greycomatrix, greycoprops
from skimage import io
```

In [118]:

```
#Read the images
D1=io.imread('D1.gif')
D2=io.imread('D2.gif')
D3=io.imread('D3.gif')
```

In [119]:

```
#Make the co-occurence matrixes
D1G = greycomatrix(D1, [1,2], [0, np.pi/4, np.pi/2, 3*np.pi/4],normed=True, symmetri
D2G = greycomatrix(D2, [1,2], [0, np.pi/4, np.pi/2, 3*np.pi/4],normed=True, symmetri
D3G = greycomatrix(D3, [1,2], [0, np.pi/4, np.pi/2, 3*np.pi/4],normed=True, symmetri
```

```
/Users/cheryl/anaconda3/lib/python3.6/site-packages/skimage/feature/te
xture.py:109: FutureWarning: Conversion of the second argument of issu
bdtype from `float` to `np.floating` is deprecated. In future, it will
be treated as `np.float64 == np.dtype(float).type`.
  if np.issubdtype(image.dtype, np.float):
```

In [120]:

```
#Hackler descriptors
D1contrast=greycoprops(D1G,'contrast')
D1correlation=greycoprops(D1G,'correlation')
D1energy=greycoprops(D1G,'energy')
D1homogeneity=greycoprops(D1G,'homogeneity')
#For D2
D2contrast=greycoprops(D2G,'contrast')
D2correlation=greycoprops(D2G,'correlation')
D2energy=greycoprops(D2G,'energy')
D2homogeneity=greycoprops(D2G,'homogeneity')
#For D3
D3contrast=greycoprops(D3G,'contrast')
D3correlation=greycoprops(D3G,'correlation')
D3energy=greycoprops(D3G,'energy')
D3homogeneity=greycoprops(D3G,'homogeneity')
```

Unpack the descriptor arrays and append them into single arrays

In [ ]:

```
#Unpack the contrast arrays
y1t=[]
for i in D1contrast[0]:
    y1t.append(i)
for i in D1contrast[1]:
    y1t.append(i)
y2t=[]
for i in D2contrast[0]:
    y2t.append(i)
for i in D2contrast[1]:
    y2t.append(i)
y3t=[]
for i in D3contrast[0]:
    y3t.append(i)
for i in D3contrast[1]:
    y3t.append(i)
#unpack the correlation arrays
y1c=[]
for i in D1correlation[0]:
    y1c.append(i)
for i in D1correlation[1]:
    y1c.append(i)
y2c=[]
for i in D2correlation[0]:
    y2c.append(i)
for i in D2correlation[1]:
    y2c.append(i)
y3c=[]
for i in D3correlation[0]:
    y3c.append(i)
for i in D3correlation[1]:
    y3c.append(i)
#Unpack the energy arrays
y1e=[]
for i in D1energy[0]:
    y1e.append(i)
for i in D1correlation[1]:
    y1e.append(i)
y2e=[]
for i in D2energy[0]:
    y2e.append(i)
for i in D2energy[1]:
    y2e.append(i)
y3e=[]
for i in D3energy[0]:
    y3e.append(i)
for i in D3energy[1]:
    y3e.append(i)
#Unpack the homogeneity arrays
```

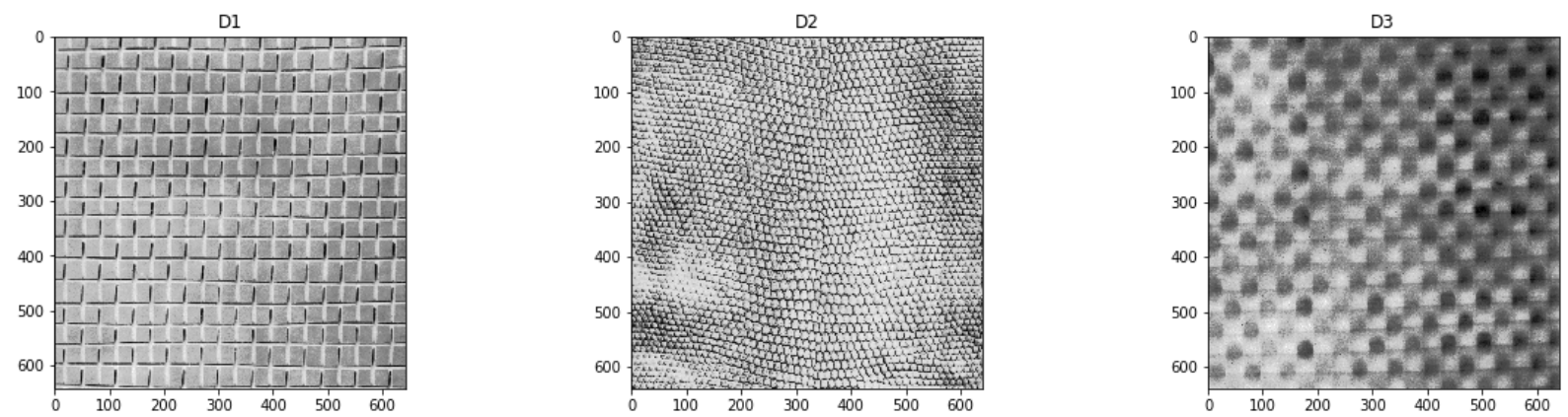
```
y1h=[]
for i in D1homogeneity[0]:
    y1h.append(i)
for i in D1homogeneity[1]:
    y1h.append(i)
y2h=[]
for i in D2homogeneity[0]:
    y2h.append(i)
for i in D2homogeneity[1]:
    y2h.append(i)
y3h=[]
for i in D3homogeneity[0]:
    y3h.append(i)
for i in D3homogeneity[1]:
    y3h.append(i)
```

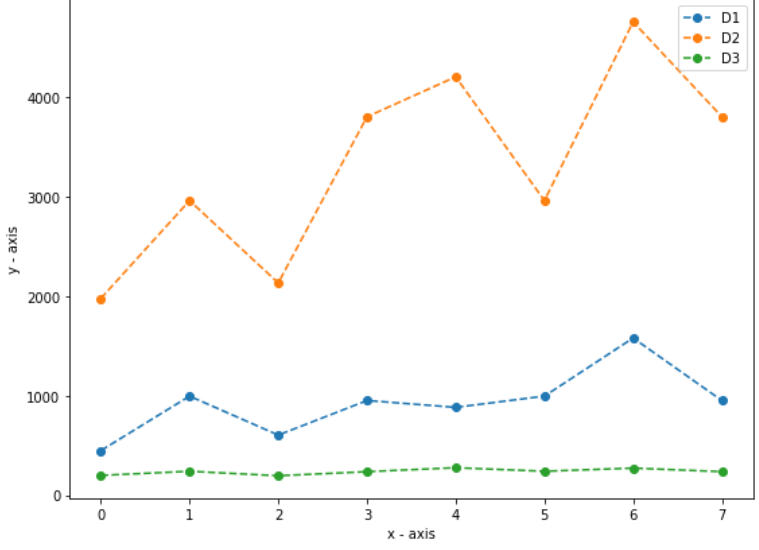
Plot out the values to compare the textures to the original images

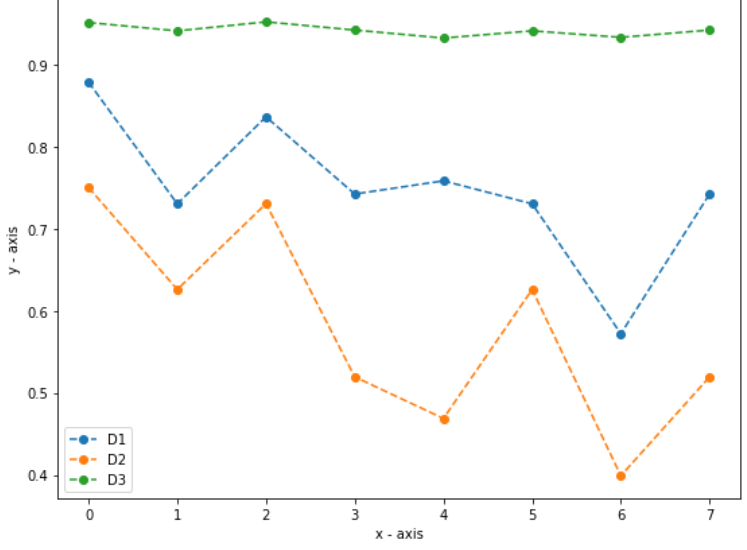
In [176]:

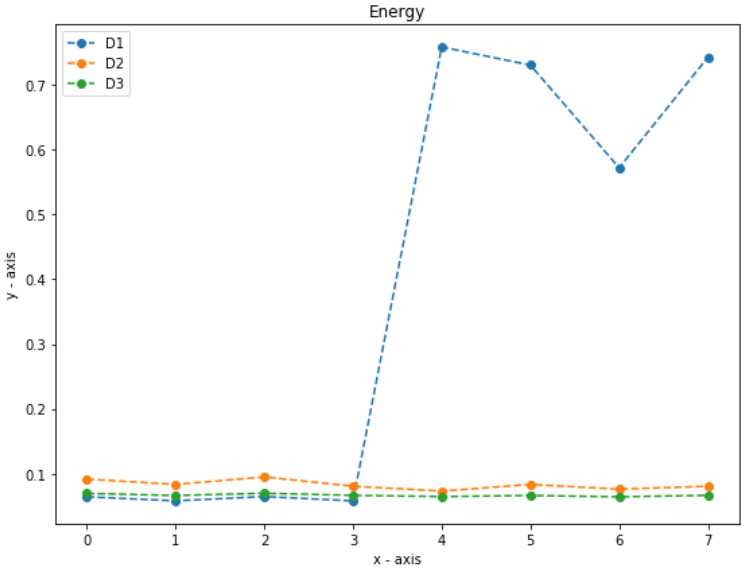
```
fig = plt.figure(figsize=(20,15))
plt.subplot(3, 3, 1)
plt.imshow(D1, cmap='gray')
plt.title("D1")
plt.subplot(3, 3, 2)
plt.imshow(D2,cmap='gray')
plt.title("D2")
plt.subplot(3, 3, 3)
plt.imshow(D3,cmap='gray')
plt.title("D3")
plt.show()
# plotting the contrast comparison
fig = plt.figure(figsize=(20,15))
plt.subplot(2, 2, 1)

plt.plot(y1t, linestyle='--', marker='o',label = "D1")
  # plotting the line 2 points
plt.plot(y2t, linestyle='--', marker='o',label = "D2")
plt.plot(y3t, linestyle='--', marker='o', label = "D3")
# naming the x axis
plt.xlabel('x - axis')
# naming the y axis
plt.ylabel('y - axis')
# giving a title to my graph
plt.title('Contast')
# show a legend on the plot
plt.legend()
  # function to show the plot
# plotting the correlation comparison
plt.subplot(2, 2, 2)
plt.plot(y1c, linestyle='--', marker='o', label = "D1")
  # plotting the line 2 points
plt.plot(y2c, linestyle='--', marker='o', label = "D2")
```

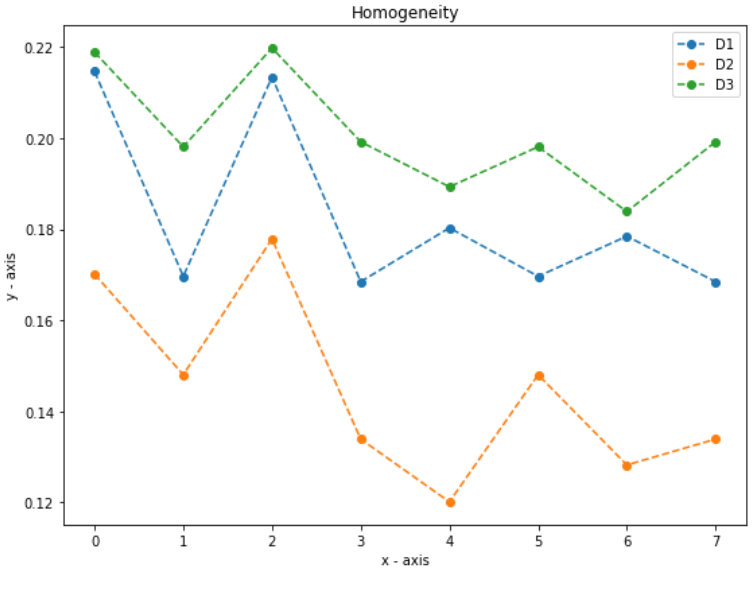
```
plt.plot(y3c, linestyle='--', marker='o', label = "D3")
# naming the x axis
plt.xlabel('x - axis')
# naming the y axis
plt.ylabel('y - axis')
# giving a title to my graph
plt.title('Correlation')
# show a legend on the plot
plt.legend()
  # function to show the plot

# plotting the energy comparison
plt.subplot(2, 2, 3)
plt.plot(y1e, linestyle='--', marker='o',label = "D1")
  # plotting the line 2 points
plt.plot(y2e,linestyle='--', marker='o', label = "D2")
plt.plot(y3e,linestyle='--', marker='o', label = "D3")
# naming the x axis
plt.xlabel('x - axis')
# naming the y axis
plt.ylabel('y - axis')
# giving a title to my graph
plt.title('Energy')
# show a legend on the plot
plt.legend()
  # function to show the plot
# plotting the homogeneity comparison
plt.subplot(2, 2, 4)
plt.plot(y1h, linestyle='--', marker='o', label = "D1")
  # plotting the line 2 points
plt.plot(y2h,  linestyle='--', marker='o',label = "D2")
plt.plot(y3h, linestyle='--', marker='o', label = "D3")
# naming the x axis
plt.xlabel('x - axis')
# naming the y axis
plt.ylabel('y - axis')
# giving a title to my graph
plt.title('Homogeneity')
# show a legend on the plot
plt.legend()
  # function to show the plot
plt.show()
```











Problem 3

Function to return the desired region properties

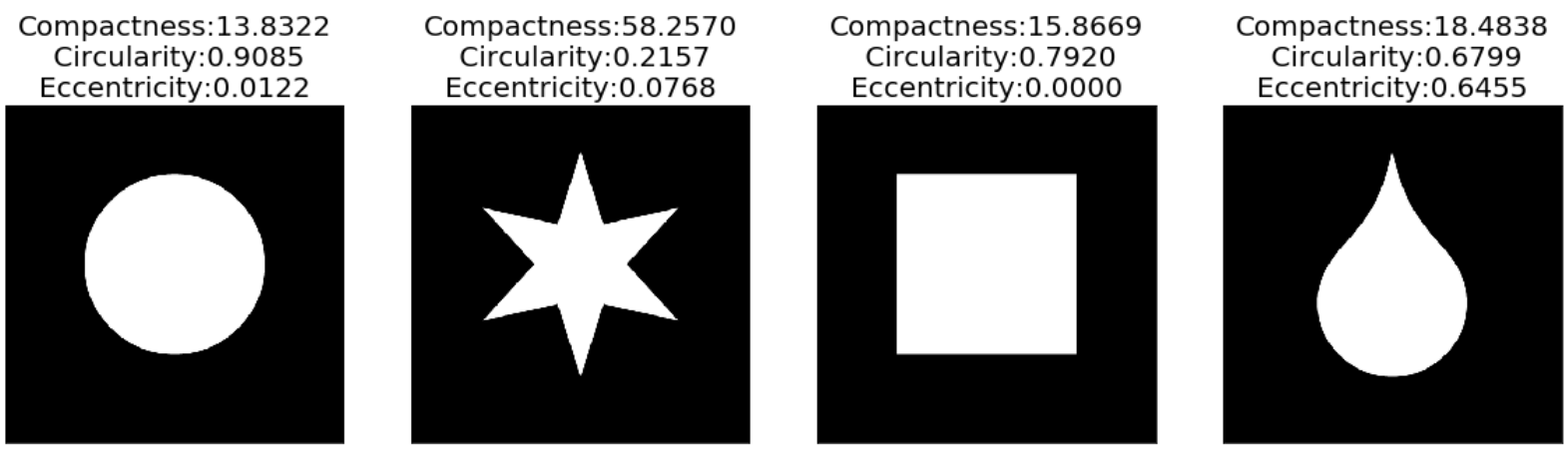
In [150]:

```
from skimage.measure import label, regionprops
import numpy as np
import skimage.io as io
# read in the image (enter the path where you downloaded it on your computer below
def regionProps(image):

    im = io.imread(image)
    bw = im> 230
    Area = np.sum(im > 230)
    regions = regionprops(bw.astype(int))
    Perimeter=regions[0].perimeter
    circularity=(4*np.pi*Area)/Perimeter**2
    compactness=Perimeter**2/(Area)
    eccentricity=regions[0].eccentricity
    return im,"{:.4f}".format(compactness), "{:.4f}".format(circularity), "{:.4f}".f
```

Execute the function to read the various shapes

In [160]:

```
circle,c1,c2,c3=regionProps('circle.tif')
star,s1,s2,s3=regionProps('star.tif')
square,q1,q2,q3=regionProps('square.tif')
tear,t1,t2,t3=regionProps('tear.tif')
plt.figure(figsize=(20,10))
plt.subplot(141)
plt.imshow(circle,cmap='gray')
plt.title('Compactness:%s\n Circularity:%s\nEccentricity:%s'%(c1,c2,c3),fontsize=20)
plt.subplot(142)
plt.imshow(star,cmap='gray')
plt.title('Compactness:%s\n Circularity:%s\nEccentricity:%s'%(s1,s2,s3),fontsize=20)
plt.subplot(143)
plt.imshow(square,cmap='gray')
plt.title('Compactness:%s\n Circularity:%s\nEccentricity:%s'%(q1,q2,q3),fontsize=20)
plt.subplot(144)
plt.imshow(tear,cmap='gray')
plt.title('Compactness:%s\n Circularity:%s\nEccentricity:%s'%(t1,t2,t3),fontsize=20)
plt.show()
```



Function to read the letters

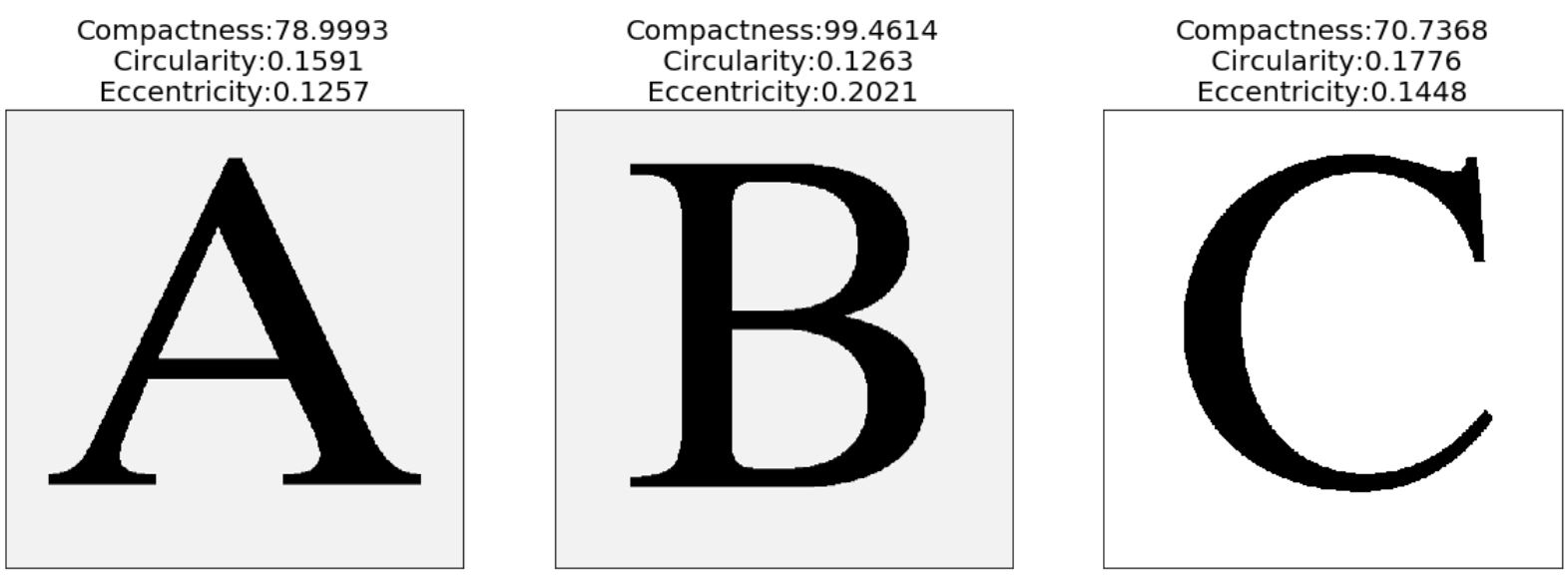
In [100]:

```
def regionProps(image):

    im = io.imread(image)
    #produces the black pixels
    bw = im< 230
    Area = np.sum(im < 230)
    regions = regionprops(bw.astype(int))
    Perimeter=regions[0].perimeter
    circularity=(4*np.pi*Area)/Perimeter**2
    compactness=Perimeter**2/(Area)
    eccentricity=regions[0].eccentricity
    return compactness, circularity, eccentricity
```

Execute the function on the letters for comparison

In [159]:

```
A,a1,a2,a3=regionProps('letterA.tif')
B,b1,b2,b3=regionProps('letterB.tif')
C,c1,c2,c3=regionProps('letterC.tif')

plt.figure(figsize=(20,10))
plt.subplot(131)
plt.imshow(A,cmap='gray')
plt.title('Compactness:%s\n Circularity:%s\nEccentricity:%s'%(a1,a2,a3),fontsize=20)
plt.subplot(132)
plt.imshow(B,cmap='gray')
plt.title('Compactness:%s\n Circularity:%s\nEccentricity:%s'%(b1,b2,b3),fontsize=20)
plt.subplot(133)
plt.imshow(C,cmap='gray')
plt.title('Compactness:%s\n Circularity:%s\nEccentricity:%s'%(c1,c2,c3),fontsize=20)

plt.show()
```



Execute the function on letter A and the distorted version for comparison

In [ ]:

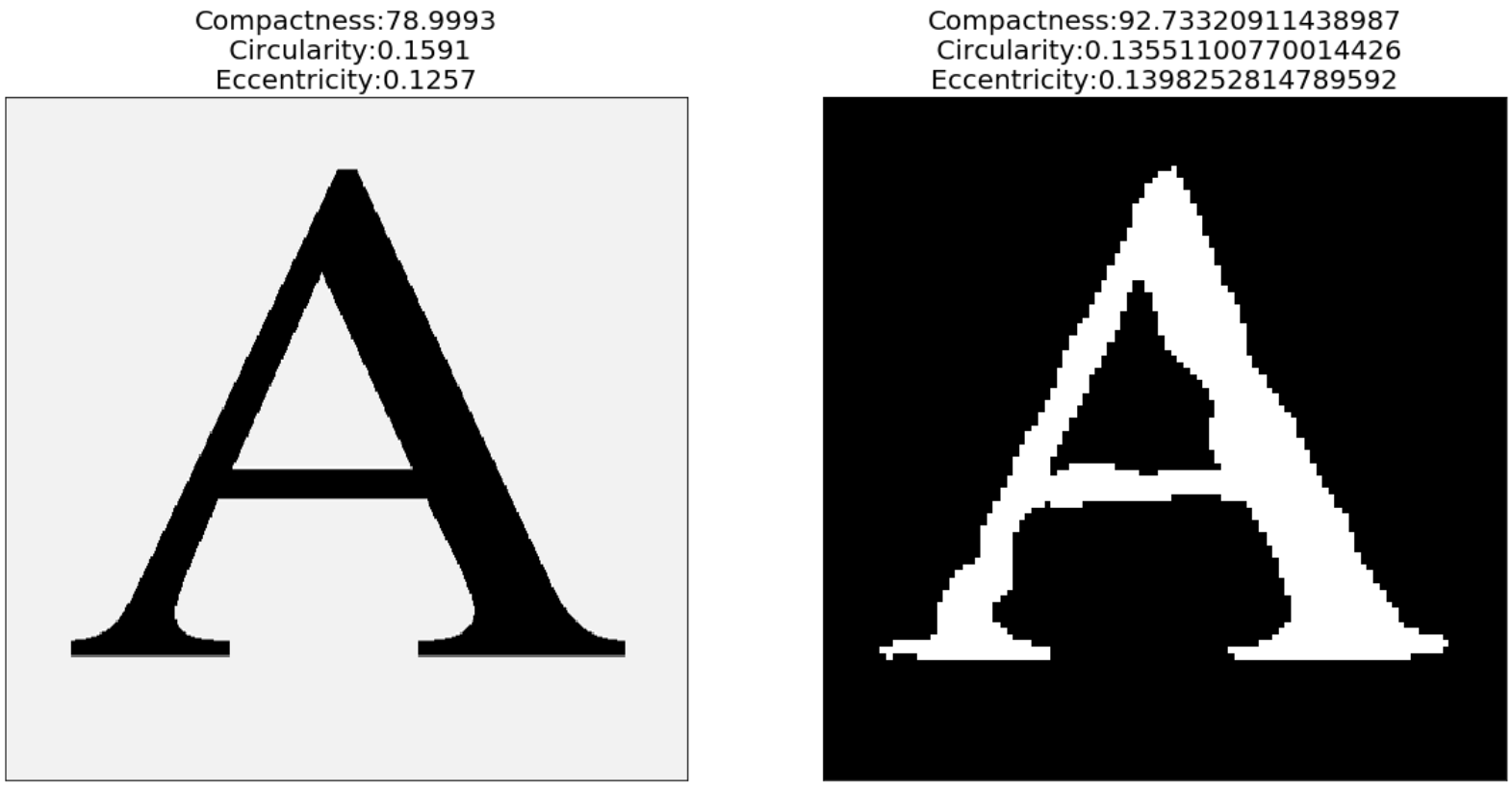
```

```

In [168]:

```
im = io.imread('letterA-distorted.tif')
bw = im==0
Area = np.sum(im == 0)
regions = regionprops(bw.astype(int))
Perimeter=regions[0].perimeter
circularity=(4*np.pi*Area)/Perimeter**2
compactness=Perimeter**2/(Area)
eccentricity=regions[0].eccentricity
```

In [169]:

```
plt.figure(figsize=(20,10))
plt.subplot(121)
plt.imshow(A,cmap='gray')
plt.title('Compactness:%s\n Circularity:%s\nEccentricity:%s'%(a1,a2,a3),fontsize=20)
plt.subplot(122)
plt.imshow(im,cmap='gray')
plt.title('Compactness:%s\n Circularity:%s\nEccentricity:%s'%(compactness,circularit
plt.show()
```



In [ ]:

Problem 2

In [119]:

In [118]:

In [119]:

```
/Users/cheryl/anaconda3/lib/python3.6/site-packages/skimage/feature/te
xture.py:109: FutureWarning: Conversion of the second argument of issu
bdtype from `float` to `np.floating` is deprecated. In future, it will
be treated as `np.float64 == np.dtype(float).type`.
  if np.issubdtype(image.dtype, np.float):
```

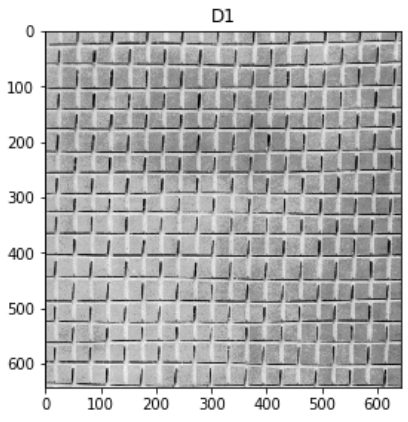
In [120]:

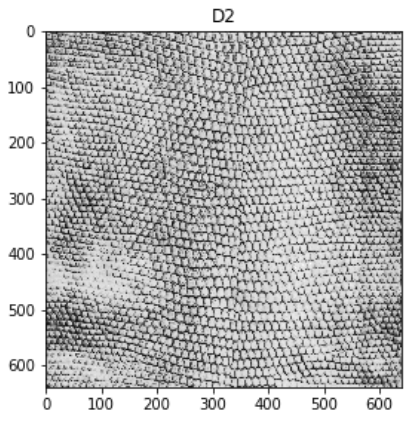
Unpack the descriptor arrays and append them into single arrays

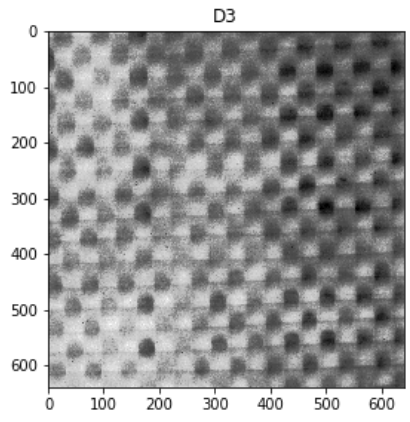
In [ ]:

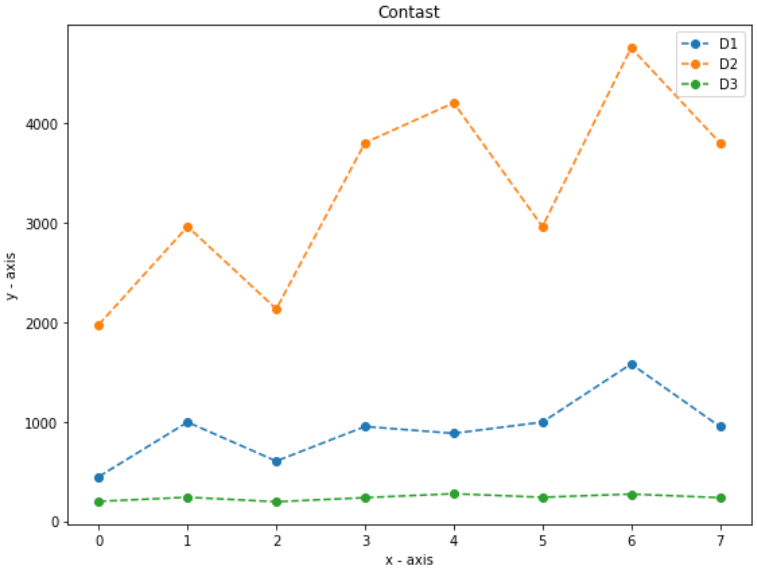
Plot out the values to compare the textures to the original images

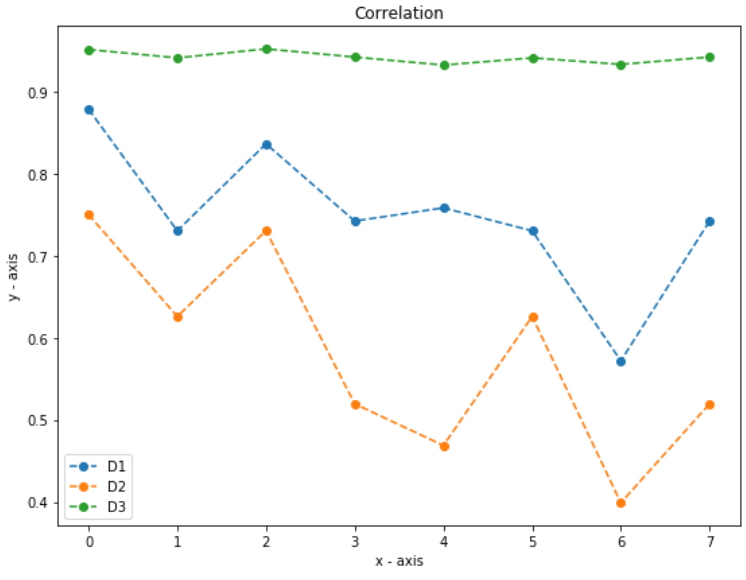
In [176]:

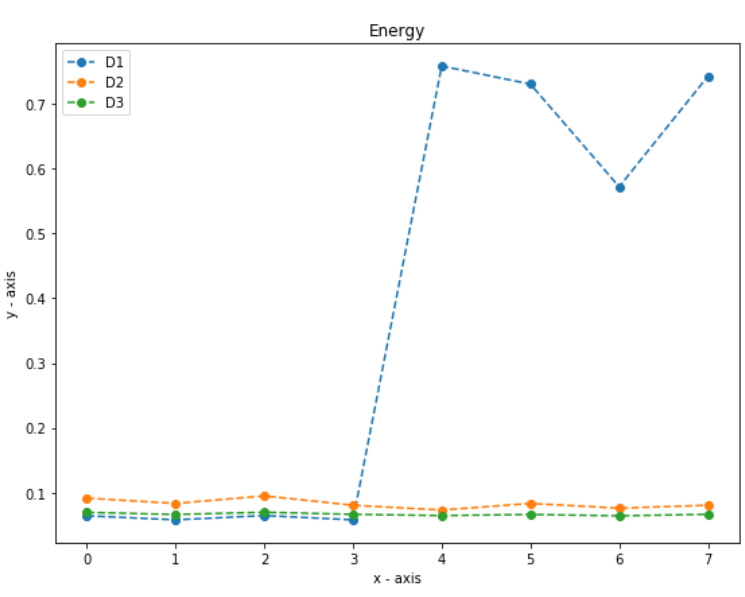


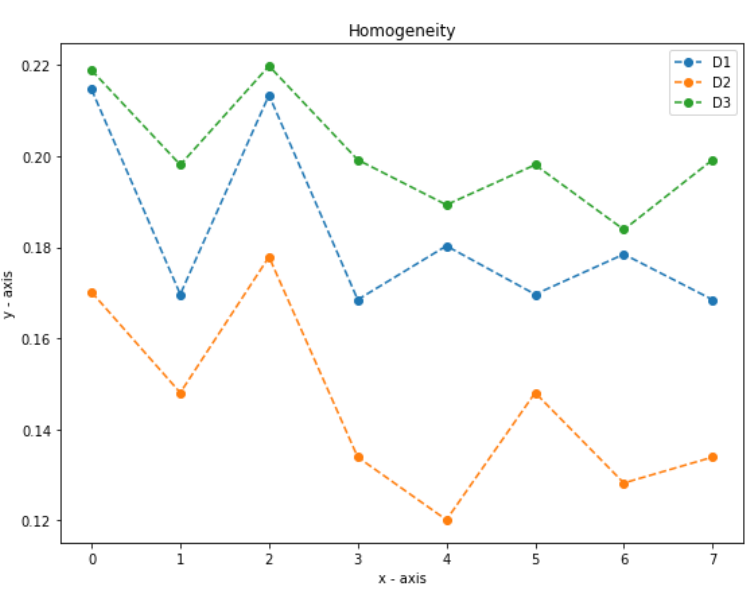












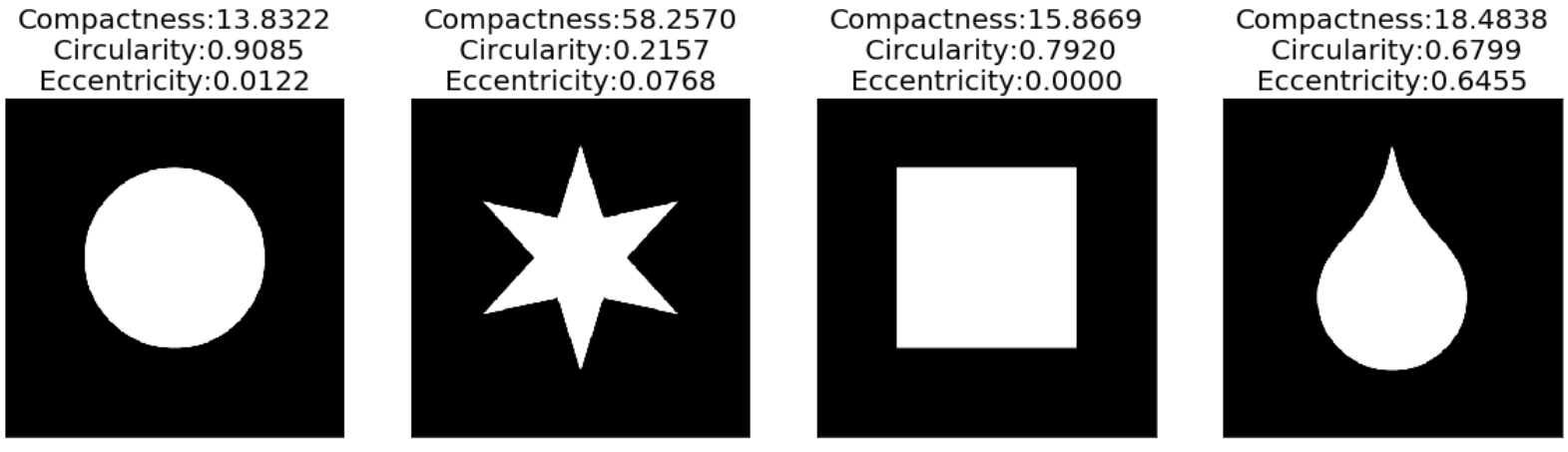
Problem 3

Function to return the desired region properties

In [150]:

Execute the function to read the various shapes

In [160]:

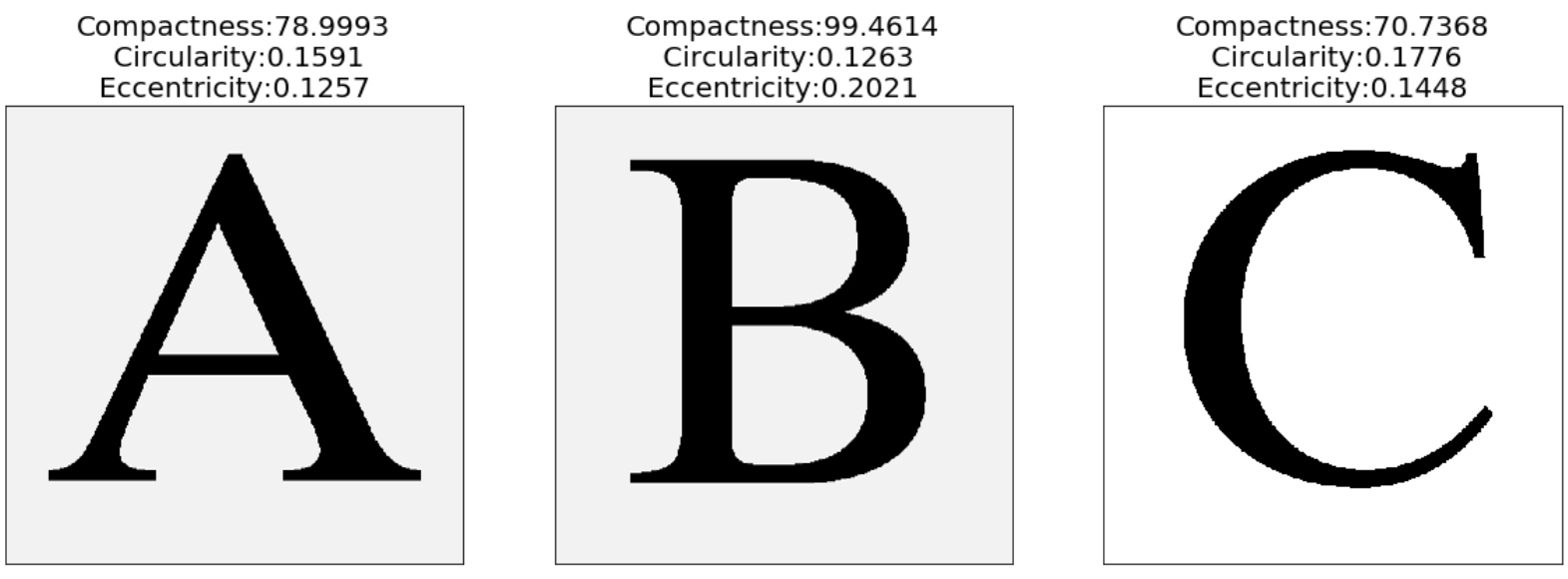


Function to read the letters

In [100]:

Execute the function on the letters for comparison

In [159]:



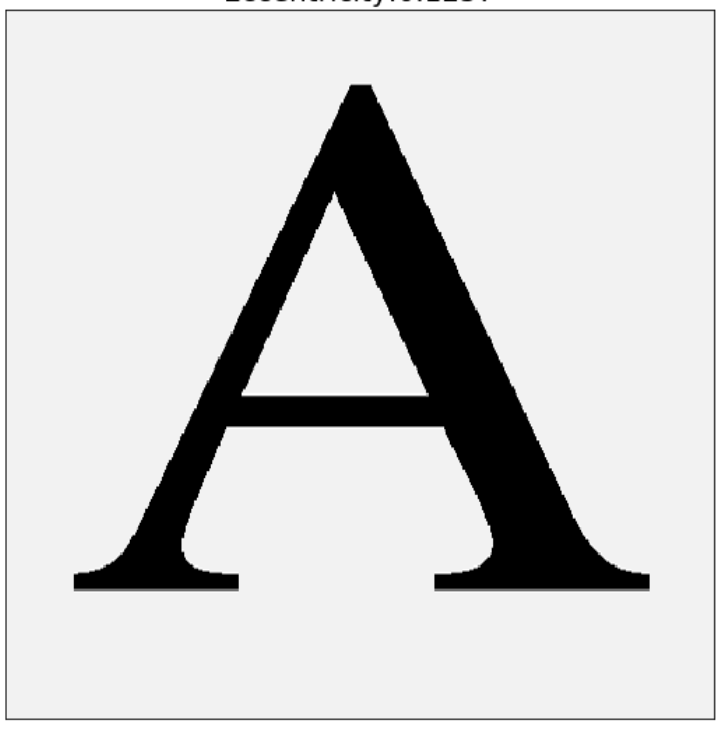
Execute the function on letter A and the distorted version for comparison

In [ ]:

In [168]:

In [169]:

Compactness:78.9993
Circularity:0.1591
Eccentricity:0.1257



Compactness:92.73320911438987
Circularity:0.13551100770014426
Eccentricity:0.1398252814789592

In [ ]: