In [3]:

```python
import cv2
import numpy as np
```

Question 2

In [4]:

```python
#part a
img = cv2.imread('1_4.bmp')
cv2.imshow('Lena', img)
cv2.waitKey(0)
cv2.destroyAllWindows()

#part b
datatype= img.dtype
print("Lena's image is data type", datatype)
img_max = np.amax(img)
print("The max is", img_max)
img_min = np.amin(img)
print("The min is", img_min)

#part c
img_double = img.astype(float)
cv2.imshow('Lena_Double', img_double)
cv2.waitKey(0)
cv2.destroyAllWindows()
#no you cannot show the double-typed image anymore

#part d
#you can show the image by converting it to uit8 data or normalizing it by dividing all values by the max data value
img_double_normalized = img_double/img_max
cv2.imshow('Lena_Double_Normalized', img_double_normalized)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
Lena's image is data type uint8
The max is 255
The min is 0
```

Question 3

In [8]:

```python
#part a
X = cv2.imread('1_2.tif') #color
Y = cv2.imread('1_2.tif', 0) #gray
cv2.imshow('X', X)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

In [6]:

```
def imRotate(X, degrees):
    height,width = X.shape
    M = cv2.getRotationMatrix2D((height/2.0,width/2.0), degrees, 1)
    return cv2.warpAffine(X,M, (height, width))
```

In [10]:

```
#part b
Z0 = imRotate(Y, 120)
cv2.imshow('Z0', Z0)
cv2.waitKey(0)
cv2.destroyAllWindows()
#part c
Z1 = Y
for x in range(12):
    Z1 = imRotate(Z1,10)
cv2.imshow('Z1', Z1)
cv2.waitKey(0)
cv2.destroyAllWindows()

#part d
#yes there is a difference, Z1 has a round black frame whereas Z0's frame is straight

#part e
```

Question 4

In [12]:

```
#part a
X = np.loadtxt('1_3.asc')
X_normalized = X / np.amax(X)
cv2.imshow('X', X_normalized)
cv2.waitKey(0)
cv2.destroyAllWindows()

#print(Y1)
Y1 = X[0:384:4,0:256:4]
Y1_normalized = Y1 / np.amax(Y1)
cv2.imshow('Y1', Y1_normalized)
cv2.waitKey(0)
cv2.destroyAllWindows()
print(X.shape)

A = X
for i in range(0, 384, 4):
    for j in range(0 , 256, 4):
        totalSum = 0
        for k in range(i, i + 3):
            for l in range(j, j + 3):
                totalSum = totalSum + X[k,l]
        A[i,j] = totalSum / 16

Y2 = A[0:384:4,0:256:4]
Y2_normalized = Y2 / np.amax(Y2)
cv2.imshow('Y2', Y2_normalized)
cv2.waitKey(0)
cv2.destroyAllWindows()

#part b
print(Y1.shape)
row, column = Y1.shape
Y1_enlarged_first = np.zeros(shape =(row * 4, column * 4))
for x in range(0,96):
    for y in range(0,64):
        for xx in range(x*4, x*4 + 4):
            for yy in range(y*4, y*4 + 4):
                Y1_enlarged_first[xx,yy] = Y1[x,y]
print(Y1_enlarged_first.shape)
cv2.imshow('Y1_enlarged_first', Y1_enlarged_first / np.amax(Y1_enlarged_first
))
cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
(384, 256)
(96, 64)
(384, 256)
```

Bi-linear Interpolation

In [13]:

```
X = np.loadtxt('1_3.asc')
Y1 = X[0:384:4,0:256:4]
Y1 = Y1 / np.amax(Y1)
cv2.imshow('Y1', Y1)
cv2.waitKey(0)
cv2.destroyAllWindows()
row, column = Y1.shape
finalColumns = column * 4
finalRows = row * 4
finalArray = np.zeros(shape=(finalRows, finalColumns))

#place values into corners
for x in range(0,96):
    for y in range(0,64):
        value = Y1[x, y]
        finalArray[x * 4, y * 4] = value
cv2.imshow('finalArray', finalArray)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

In [14]:

```
#interpolate across the x
for i in range(0, 384, 4):
    for j in range(0, 252, 4):
        firstValue = (3/4) * finalArray[i, j] + (1/4) * finalArray[i, j+4]
        secondValue = (1/2) * finalArray[i, j] + (2/4) * finalArray[i, j+4]
        thirdValue = (1/4) * finalArray[i, j] + (3/4) * finalArray[i, j+4]
        finalArray[i, j + 1] = firstValue
        finalArray[i, j + 2] = secondValue
        finalArray[i, j + 3] = thirdValue
cv2.imshow('finalArray', finalArray)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

In [15]:

```
#interpolate across the y
for jj in range(0, 256, 4):
    for ii in range(0, 380, 4):
        firstValue = (3/4) * finalArray[ii, jj] + (1/4) * finalArray[ii + 4, j
j]
        secondValue = (1/2) * finalArray[ii, jj] + (1/2) * finalArray[ii + 4,
jj]
        thirdValue = (1/4) * finalArray[ii, jj] + (3/4) * finalArray[ii + 4, j
j]
        finalArray[ii + 1, jj] = firstValue
        finalArray[ii + 2, jj] = secondValue
        finalArray[ii + 3, jj] = thirdValue
cv2.imshow('finalArray', finalArray)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

In [16]:

```
#now do 3x3 panel
for iii in range(0, 380, 4):
    for jjj in range(0, 252, 4):
        for x in range (1, 4):
            for y in range (1, 4):
                firstTerm = (4 - x)/4 * (4 - y)/4 * finalArray[iii, jjj]
                secondTerm = x/4 * (4 - y)/4 * finalArray[iii + 4, jjj]
                thirdTerm = (4 - x)/4 * y/4 * finalArray[iii, jjj + 4]
                fourthTerm = x/4 * y/4 * finalArray[iii + 4, jjj + 4]
                finalArray[iii + x, jjj + y] = firstTerm + secondTerm + thirdT
erm + fourthTerm
cv2.imshow('finalArray', finalArray)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

In [17]:

```
#right side and bottom
for row in range(0, 384, 1):
    finalArray[row, 252] = finalArray[row, 251]
    finalArray[row, 253] = finalArray[row, 251]
    finalArray[row, 254] = finalArray[row, 251]
    finalArray[row, 255] = finalArray[row, 251]
for col in range(0, 256, 1):
    finalArray[380, col] = finalArray[379, col]
    finalArray[381, col] = finalArray[379, col]
    finalArray[382, col] = finalArray[379, col]
    finalArray[383, col] = finalArray[379, col]
cv2.imshow('finalArray', finalArray)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

In [ ]:

In [ ]: