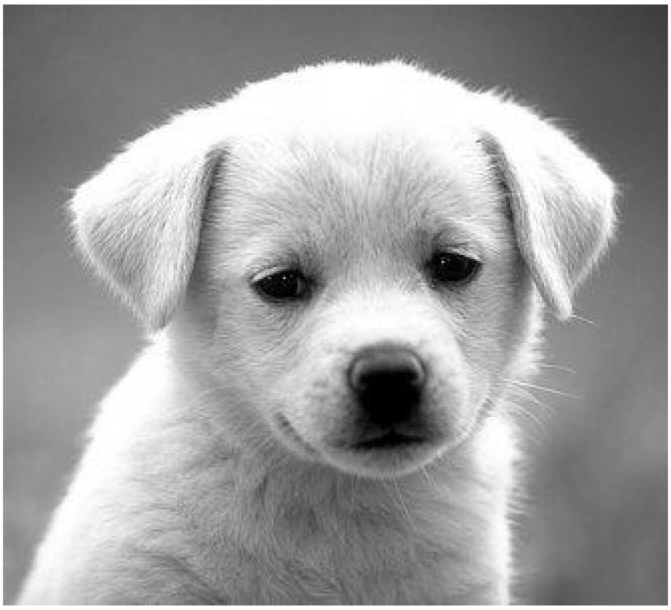
(A)

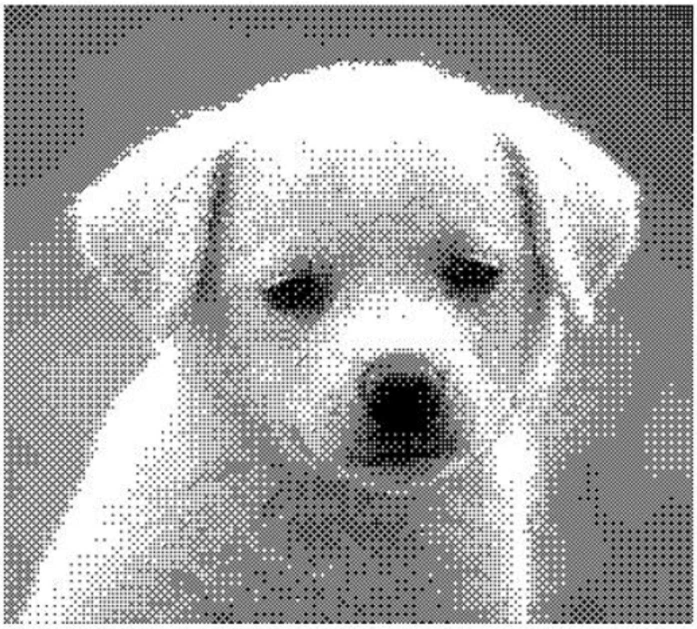
source code:

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import cv2

# process thresholdong
def thresholding(input_img):
    # initialize dithering matrix
    D = [[0, 128, 32, 160],
         [192, 64, 224, 96],
         [48, 176, 16, 144],
         [240, 112, 208, 80]]

    # get image's x and y
    y = len(input_img)
    x = len(input_img[0])

    for i in range(y):
        for j in range(x):
            # check
            if input_img[i][j][0] > D[i % 4][j % 4]:
                change = 255
            else:
                change = 0

            # change color
            for k in range(3):
                input_img[i][j][k] = change

# read the image
img = mpimg.imread("input.jpg")

print('\nshowing origin image...')

plt.imshow(img)
# disable axis
plt.axis('off')
# show the image
plt.show()

print('\nPocessing...\n ')

thresholding(img)

plt.imshow(img)
plt.axis('off')
# save imgage's snapshot
plt.savefig('output.jpg')
#show the grayscale image
plt.show()
```

origin image:



processed image:



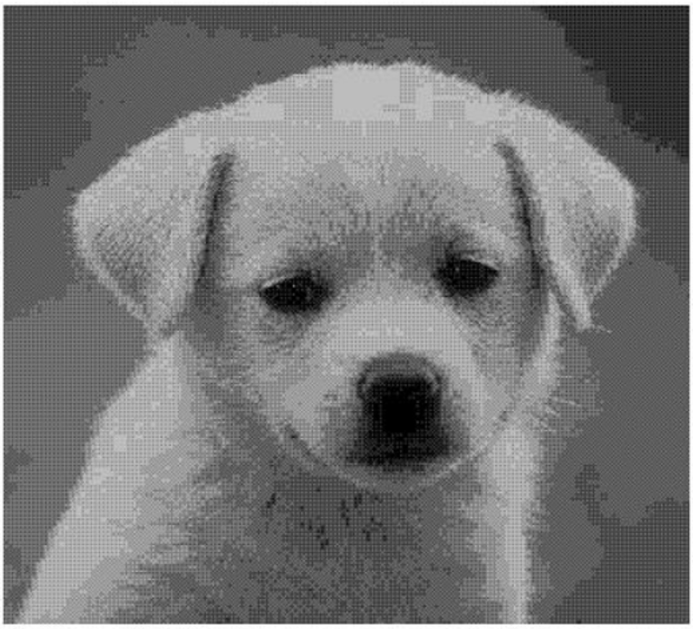
(B)

source code:

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import cv2

# process extend to n = 4 gray scale
 # Q = I / 85
def preprocess(input_img):
    # get image's x and y
    y = len(input_img)
    x = len(input_img[0])

    for i in range(y):
        for j in range(x):
            for k in range(3):
                input_img[i][j][k] //= 85
 # check
def extend_n4(I, Q):
    # initialize dithering matrix
    D = [[0, 56],
         [84, 28]]

    # get image's x and y
    y = len(I)
    x = len(I[0])

    for i in range(y):
        for j in range(x):
            # check
            if I[i][j][0] - (85 * Q[i][j][0]) > D[i % 2][j % 2]:
                change = 1
            else:
                change = 0

            # change color
            for k in range(3):
                I[i][j][k] = (Q[i][j][k] + change) * 63

# read the image
img = mpimg.imread("input.jpg")
cpy = img.copy()

print('\nshowing origin image...')

plt.imshow(img)
# disable axis
plt.axis('off')
# show the image
plt.show()

print('\nPocessing...\n ')

# get Q
preprocess(cpy)

print(cpy)
extend_n4(img, cpy)

plt.imshow(img)
plt.axis('off')
# save imgage's snapshot
plt.savefig('output_n4_gray_values.jpg')
#show the grayscale image
plt.show()
```

origin image:

processed image:



心得：

這次的作業蠻好玩的，寫完之後在網路上有查到很多更厲害的寫法，發現 numpy 真是個強大的工具，之後再試著用用看，這次的作業就先用暴力法帶過了。