

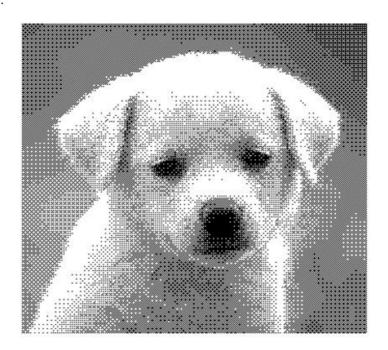
source code:

```
1
    import numpy as np
 2
    import matplotlib.pyplot as plt
 3
    import matplotlib.image as mpimg
 4
    import cv2
 5
 6
    # process thresholdong
 7
    def thresholding(input_img):
 8
        # initialize dithering matrix
        D = [[0, 128, 32, 160],
 9
10
             [192, 64, 224, 96],
11
             [48, 176, 16, 144],
12
             [240, 112, 208, 80]]
13
14
        # get image's x and y
15
        y = len(input_img)
        x = len(input_img[0])
16
17
        for i in range(y):
18
19
            for j in range(x):
                 # check
20
21
                 if input_img[i][j][0] > D[i % 4][j % 4]:
22
                     change = 255
23
                 else:
24
                     change = 0
25
26
                 # change color
27
                for k in range(3):
28
                     input_img[i][j][k] = change
29
30
    # read the image
    img = mpimg.imread("input.jpg")
31
32
33
    print('\nshowing origin image...')
34
35
    plt.imshow(img)
    # disable axis
36
37
    plt.axis('off')
38
    # show the image
39
    plt.show()
40
    print('\nPocessing...\n ')
41
42
43
    thresholding(img)
44
45
    plt.imshow(img)
46
    plt.axis('off')
    # save imgage's snapshot
47
48
    plt.savefig('output.jpg')
49
    #show the grayscale image
50
    plt.show()
```

origin image:



processed image:



(B)

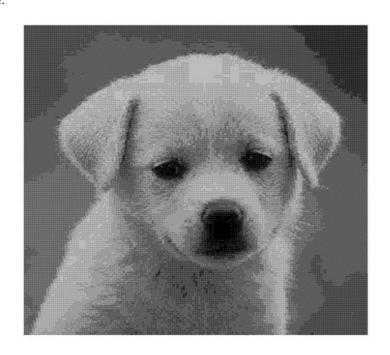
source code:

```
import numpy as np
    import matplotlib.pyplot as plt
3
    import matplotlib.image as mpimg
    import cv2
5
    \# process extend to n = 4 gray scale
6
    \# Q = I / 85
8
    def preprocess(input_img):
9
        # get image's x and y
10
        y = len(input_img)
        x = len(input_img[0])
11
12
        for i in range(y):
13
```

```
14
            for j in range(x):
15
                for k in range(3):
16
                    input_img[i][j][k] //= 85
17
     # check
18
    def extend_n4(I, Q):
19
        # initialize dithering matrix
20
        D = [[0, 56],
21
             [84, 28]]
22
23
        # get image's x and y
24
        y = len(I)
25
        x = len(I[0])
26
27
        for i in range(y):
28
            for j in range(x):
29
                # check
30
                if I[i][j][0] - (85 * Q[i][j][0]) > D[i % 2][j % 2]:
31
                    change = 1
32
                else:
33
                    change = 0
34
                # change color
35
36
                for k in range(3):
37
                    I[i][j][k] = (Q[i][j][k] + change) * 63
38
    # read the image
39
40
    img = mpimg.imread("input.jpg")
41
    cpy = img.copy()
42
43
    print('\nshowing origin image...')
44
45
    plt.imshow(img)
46 # disable axis
47 plt.axis('off')
48
    # show the image
49
    plt.show()
50
51
    print('\nPocessing...\n ')
52
53
    # get Q
54
    preprocess(cpy)
55
56
    print(cpy)
57
    extend_n4(img, cpy)
58
59
    plt.imshow(img)
60
    plt.axis('off')
61 | # save imgage's snapshot
62 plt.savefig('output_n4_gray_values.jpg')
63
    #show the grayscale image
    plt.show()
```



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。