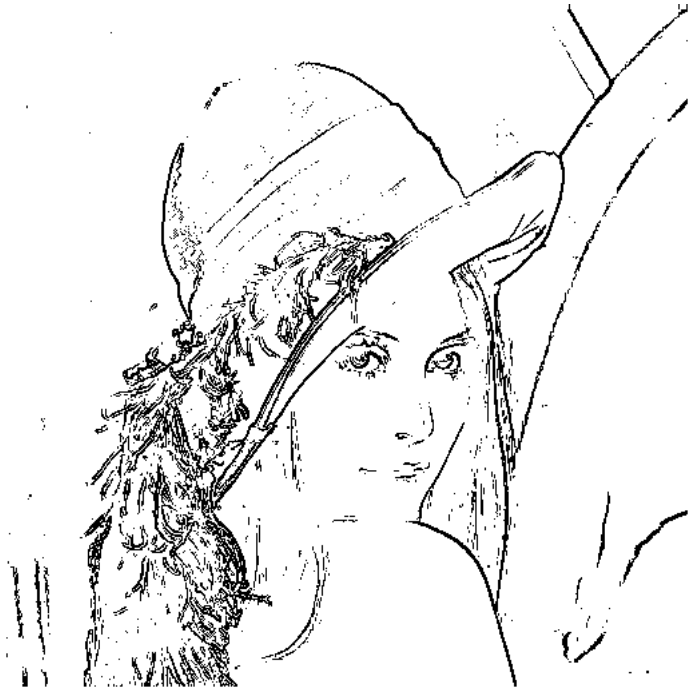
Howework 9 Report

Student:鞠之浩 ID:P10942A08

Output:

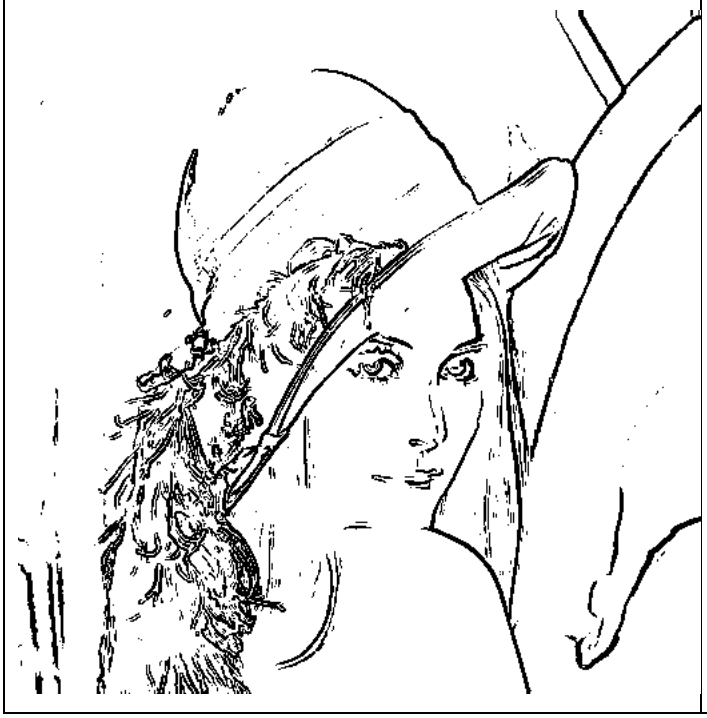
Part(a)

Robert's Operator, threshold: 30



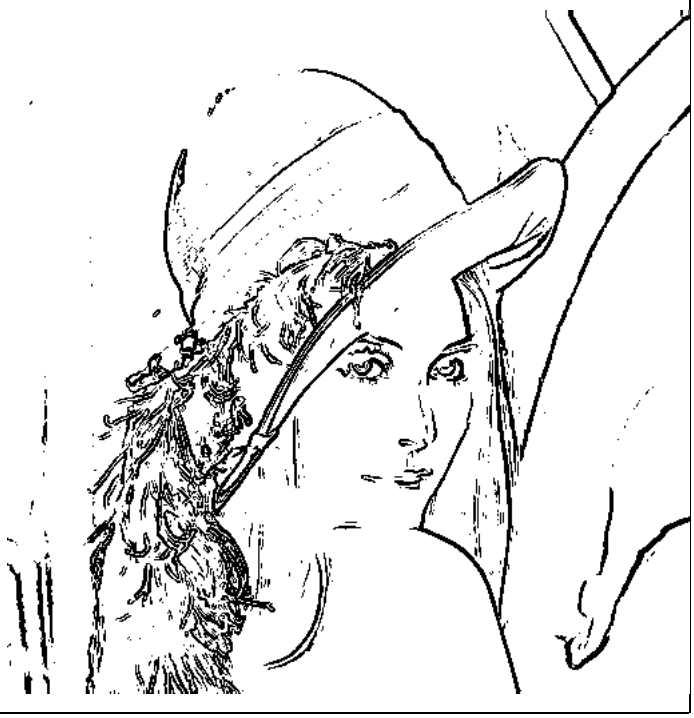
Part(b)

Prewitt's Edge Detector, threshold: 90



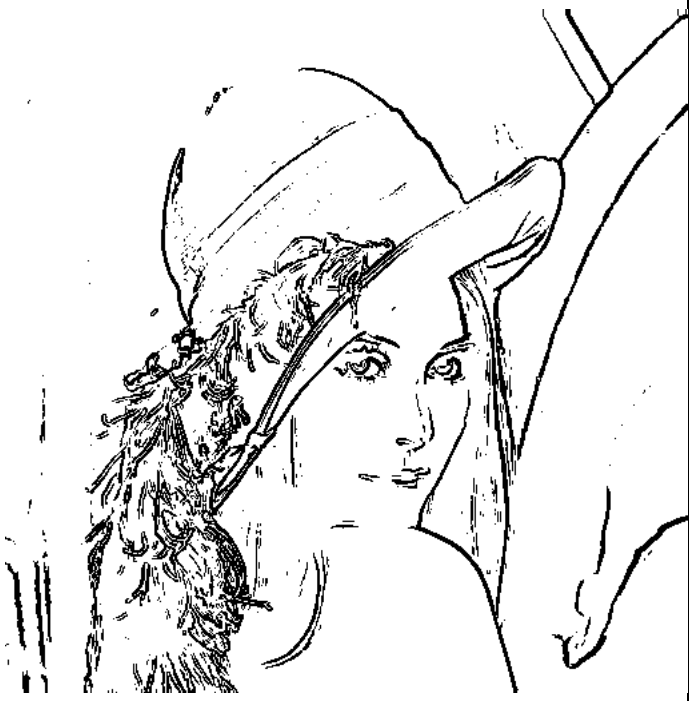
Part(c)

Sobel's Edge Detector, threshold: 120



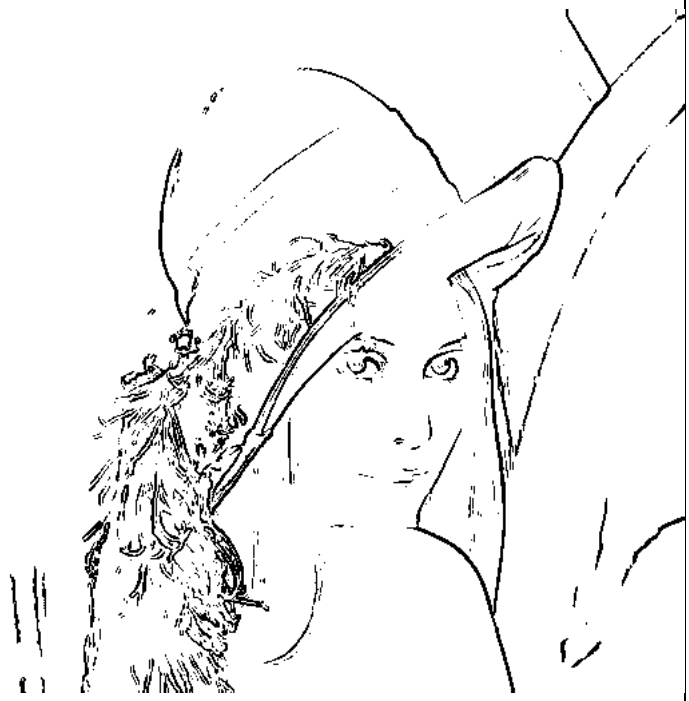
Part(d)

Frei and Chen's Gradient Operator, threshold: 103



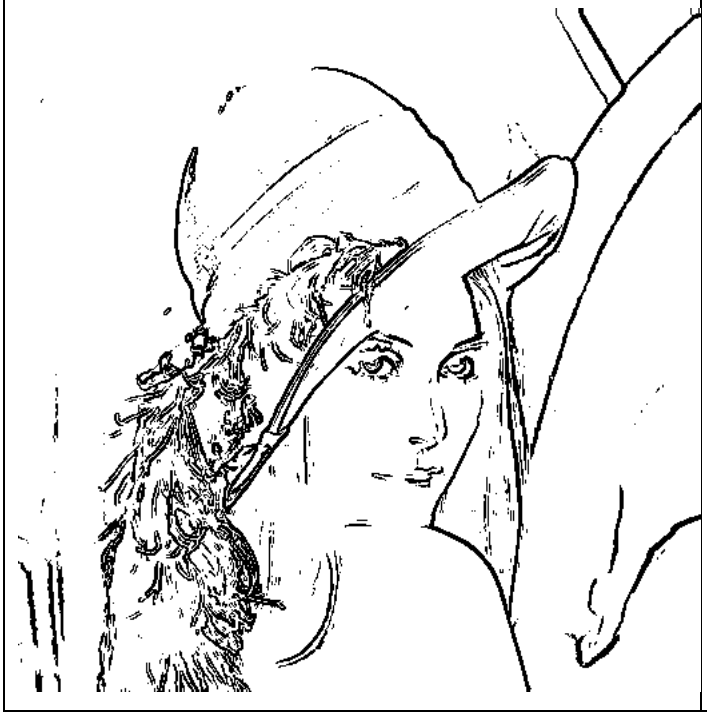
Part(e)

Kirsch's Compass Operator, threshold: 500



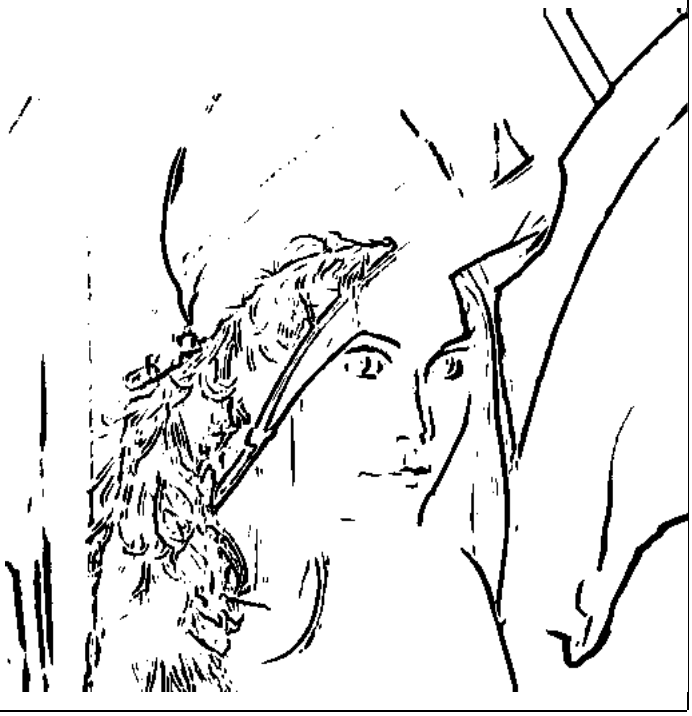
Part(f)

Robinson's Compass Operator, threshold: 123



Part(g)

Nevatia-Babu 5x5 Operator, threshold: 32100



Code describe:

Part(a):

```
def Roberts_Operator(img,threshold):
    answer = img.copy()
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            r1 = 0
            r2 = 0
            if i+1 < img.shape[0] and j+1 < img.shape[1]:
                r1 = int(img[i+1,j+1]) - int(img[i,j])
                r2 = int(img[i+1,j]) - int(img[i,j+1])
            if math.sqrt(r1*r1+r2*r2) >= threshold:
                answer[i,j] = 0
            else:
                answer[i,j] = 255
    return answer
```

do Robert's_Operator for every pixel.

```
def padding(img,size):
    answer = cv2.copyMakeBorder(img, size//2, size//2, size//2, size//2, cv2.BORDER_REFLECT)
    return answer
```

Part(b):

```
def Prewitts_Edge_Detector(img,threshold):
    answer = img.copy()
    pad = padding(img,3)
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            p1 = 0
            p2 = 0
            up = 0
            down = 0
            left = 0
            right = 0
            up = int(pad[i,j])+int(pad[i,j+1])+int(pad[i,j+2])
            down = int(pad[i+2][j])+int(pad[i+2][j+1])+int(pad[i+2][j+2])
            left = int(pad[i,j])+int(pad[i+1,j])+int(pad[i+2,j])
            right = int(pad[i,j+2])+int(pad[i+1,j+2])+int(pad[i+2,j+2])
            p1 = down - up
            p2 = right - left
            if math.sqrt(p1*p1+p2*p2) >= threshold:
                answer[i,j] = 0
            else:
                answer[i,j] = 255
    return answer
```

do padding first, then do Prewitt's_Edge_Detector for every pixel.

Part(c):

```
def Sobels_Edge_Detector(img,threshold):
    answer = img.copy()
    pad = padding(img,3)
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            p1 = 0
            p2 = 0
            up = 0
            down = 0
            left = 0
            right = 0
            up = int(pad[i,j])+2*int(pad[i,j+1])+int(pad[i,j+2])
            down = int(pad[i+2][j])+2*int(pad[i+2][j+1])+int(pad[i+2][j+2])
            left = int(pad[i,j])+2*int(pad[i+1,j])+int(pad[i+2,j])
            right = int(pad[i,j+2])+2*int(pad[i+1,j+2])+int(pad[i+2,j+2])
            p1 = down - up
            p2 = right - left
            if math.sqrt(p1*p1+p2*p2) >= threshold:
                answer[i,j] = 0
            else:
                answer[i,j] = 255
    return answer
```

do padding first, then do Sobel's_Edge_Detector for every pixel.

Part(d):

```
def Frei_and_Chens_Gradient_Operator(img,threshold):
    answer = img.copy()
    pad = padding(img,3)
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            p1 = 0
            p2 = 0
            up = 0
            down = 0
            left = 0
            right = 0
            up = int(pad[i,j])+math.sqrt(2)*int(pad[i,j+1])+int(pad[i,j+2])
            down = int(pad[i+2][j])+math.sqrt(2)*int(pad[i+2][j+1])+int(pad[i+2][j+2])
            left = int(pad[i,j])+math.sqrt(2)*int(pad[i+1,j])+int(pad[i+2,j])
            right = int(pad[i,j+2])+math.sqrt(2)*int(pad[i+1,j+2])+int(pad[i+2,j+2])
            p1 = down - up
            p2 = right - left
            if math.sqrt(p1*p1+p2*p2) >= threshold:
                answer[i,j] = 0
            else:
                answer[i,j] = 255
    return answer
```

do padding first, then do Frei_and_Chen's_Gradient_Operator for every pixel.

```
def Kirschs_Compass_Operator(img,threshold):
    answer = img.copy()
    pad = padding(img,3)
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            k = []
            for l in range(8):
                #print(l)
                if l == 0:
                    k.append(5*(int(pad[i+2,j+2])+int(pad[i+1,j+2])+int(pad[i,j+2]))
                             -3*(int(pad[i,j+1])+int(pad[i,j])+int(pad[i+1,j])+int(pad[i+2,j])+int(pad[i+2,j+1])))
                elif l == 1:
                    k.append(5*(int(pad[i+1,j+2])+int(pad[i,j+2])+int(pad[i,j+1]))
                             -3*(int(pad[i,j])+int(pad[i+1,j])+int(pad[i+2,j])+int(pad[i+2,j+1])+int(pad[i+2,j+2])))
                elif l == 2:
                    k.append(5*(int(pad[i,j+2])+int(pad[i,j+1])+int(pad[i,j]))
                             -3*(int(pad[i+1,j])+int(pad[i+2,j])+int(pad[i+2,j+1])+int(pad[i+2,j+2])+int(pad[i+1,j+2])))
                elif l == 3:
                    k.append(5*(int(pad[i,j+1])+int(pad[i,j])+int(pad[i+1,j]))
                             -3*(int(pad[i+2,j])+int(pad[i+2,j+1])+int(pad[i+2,j+2])+int(pad[i+1,j+2])+int(pad[i,j+2])))
                elif l == 4:
                    k.append(5*(int(pad[i,j])+int(pad[i+1,j])+int(pad[i+2,j]))
                             -3*(int(pad[i+2,j+1])+int(pad[i+2,j+2])+int(pad[i+1,j+2])+int(pad[i,j+2])+int(pad[i,j+1])))
                elif l == 5:
                    k.append(5*(int(pad[i+1,j])+int(pad[i+2,j])+int(pad[i+2,j+1]))
                             -3*(int(pad[i+2,j+2])+int(pad[i+1,j+2])+int(pad[i,j+2])+int(pad[i,j+1])+int(pad[i,j])))
                elif l == 6:
                    k.append(5*(int(pad[i+2,j])+int(pad[i+2,j+1])+int(pad[i+2,j+2]))
                             -3*(int(pad[i+1,j+2])+int(pad[i,j+2])+int(pad[i,j+1])+int(pad[i,j])+int(pad[i+1,j])))
                elif l == 7:
                    k.append(5*(int(pad[i+2,j+1])+int(pad[i+2,j+2])+int(pad[i+1,j+2]))
                             -3*(int(pad[i,j+2])+int(pad[i,j+1])+int(pad[i,j])+int(pad[i+1,j])+int(pad[i+2,j])))
            k = np.array(k)
            if max(k) >= threshold:
                answer[i,j] = 0
            else:
                answer[i,j] = 255
    return answer
```

Part(e): do padding first, then do Kirsch's_Compass_Operator for every pixel.

```
def Robinsons_Compass_Operator(img,threshold):
    answer = img.copy()
    pad = padding(img,3)
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            r = []
            for l in range(8):
                if l == 0:
                    r.append(int(pad[i+2,j+2])+2*int(pad[i+1,j+2])+int(pad[i,j+2])
                             -int(pad[i,j])-2*int(pad[i+1,j])-int(pad[i+2,j]))
                elif l == 1:
                    r.append(int(pad[i+1,j+2])+2*int(pad[i,j+2])+int(pad[i,j+1])
                             -int(pad[i+1,j])-2*int(pad[i+2,j])-int(pad[i+2,j+1]))
                elif l == 2:
                    r.append(int(pad[i,j])+2*int(pad[i,j+1])+int(pad[i,j+2])
                             -int(pad[i+2,j])-2*int(pad[i+2,j+1])-int(pad[i+2,j+2]))
                elif l == 3:
                    r.append(int(pad[i,j+1])+2*int(pad[i,j])+int(pad[i+1,j])
                             -int(pad[i+2,j+1])-2*int(pad[i+2,j+2])-int(pad[i+1,j+2]))
                elif l == 4:
                    r.append(int(pad[i,j])+2*int(pad[i+1,j])+int(pad[i+2,j])
                             -int(pad[i,j+2])-2*int(pad[i+1,j+2])-int(pad[i+2,j+2]))
                elif l == 5:
                    r.append(int(pad[i+1,j])+2*int(pad[i+2,j])+int(pad[i+2,j+1])
                             -int(pad[i,j+1])-2*int(pad[i,j+2])-int(pad[i+1,j+2]))
                elif l == 6:
                    r.append(int(pad[i+2,j])+2*int(pad[i+2,j+1])+int(pad[i+2,j+2])
                             -int(pad[i,j])-2*int(pad[i,j+1])-int(pad[i,j+2]))
                elif l == 7:
                    r.append(int(pad[i+2,j+1])+2*int(pad[i+2,j+2])+int(pad[i+1,j+2])
                             -int(pad[i,j+1])-2*int(pad[i,j])-int(pad[i+1,j]))
            r = np.array(r)
            if max(r) >= threshold:
                answer[i,j] = 0
            else:
                answer[i,j] = 255
    return answer
```

Part(f): do padding first, then do Robinson's_Compass_Operator for every pixel.

```
Nevatia_Babu5X5_Operator(img,threshold):
answer = img.copy()
pad = padding(img,5)
for i in range(img.shape[0]):
    for j in range(img.shape[1]):
        N = []
        for l in range(6):
            if l == 0:
                N.append(100*(int(pad[i,j])+int(pad[i,j+1])+int(pad[i,j+2])+int(pad[i,j+3])+int(pad[i,j+4])
                         +int(pad[i+1,j])+int(pad[i+1,j+1])+int(pad[i+1,j+2])+int(pad[i+1,j+3])+int(pad[i+1,j+4]))
                        -100*(int(pad[i+3,j])+int(pad[i+3,j+1])+int(pad[i+3,j+2])+int(pad[i+3,j+3])+int(pad[i+3,j+4])
                         +int(pad[i+4,j])+int(pad[i+4,j+1])+int(pad[i+4,j+2])+int(pad[i+4,j+3])+int(pad[i+4,j+4]))
                        )
            elif l == 1:
                N.append(100*(int(pad[i,j])+int(pad[i,j+1])+int(pad[i,j+2])+int(pad[i,j+3])+int(pad[i,j+4])
                         +int(pad[i+1,j])+int(pad[i+1,j+1])+int(pad[i+1,j+2])+int(pad[i+2,j]))
                        +92*(int(pad[i+2,j+1]))-92*(int(pad[i+2,j+3]))
                        +78*(int(pad[i+1,j+3]))-78*(int(pad[i+3,j+1]))
                        +32*(int(pad[i+3,j]))-32*(int(pad[i+1,j+4]))
                        -100*(int(pad[i+2,j+4])+int(pad[i+3,j+2])+int(pad[i+3,j+3])+int(pad[i+3,j+4])
                         +int(pad[i+4,j])+int(pad[i+4,j+1])+int(pad[i+4,j+2])+int(pad[i+4,j+3])+int(pad[i+4,j+4]))
                        )
            elif l == 2:
                N.append(100*(int(pad[i,j])+int(pad[i,j+1])+int(pad[i,j+2])+int(pad[i+1,j])+int(pad[i+1,j+1])
                         +int(pad[i+2,j])+int(pad[i+2,j+1])+int(pad[i+3,j])+int(pad[i+4,j]))
                        +92*(int(pad[i+1,j+2]))-92*(int(pad[i+3,j+2]))
                        +78*(int(pad[i+3,j+1]))-78*(int(pad[i+1,j+3]))
                        +32*(int(pad[i,j+3]))-32*(int(pad[i+1,j+4]))
                        -100*(int(pad[i,j+4])+int(pad[i+1,j+3])+int(pad[i+2,j+3])+int(pad[i+2,j+4])+int(pad[i+3,j+3])
                         +int(pad[i+3,j+4])+int(pad[i+4,j+2])+int(pad[i+4,j+3])+int(pad[i+4,j+4]))
                        )
            elif l == 3:
                N.append(100*(int(pad[i,j+3])+int(pad[i+1,j+3])+int(pad[i+2,j+3])+int(pad[i+3,j+3])+int(pad[i+4,j+3])
                         +int(pad[i,j+4])+int(pad[i+1,j+4])+int(pad[i+2,j+4])+int(pad[i+3,j+4])+int(pad[i+4,j+4]))
                        -100*(int(pad[i,j])+int(pad[i+1,j])+int(pad[i+2,j])+int(pad[i+3,j])+int(pad[i+4,j])
                         +int(pad[i,j+1])+int(pad[i+1,j+1])+int(pad[i+2,j+1])+int(pad[i+3,j+1])+int(pad[i+4,j+1]))
                        )
            elif l == 4:
                N.append(100*(int(pad[i,j+2])+int(pad[i,j+3])+int(pad[i,j+4])+int(pad[i+1,j+3])+int(pad[i+1,j+4])
                         +int(pad[i+2,j+3])+int(pad[i+2,j+4])+int(pad[i+3,j+4])+int(pad[i+4,j+4]))
                        +92*(int(pad[i+1,j+2]))-92*(int(pad[i+3,j+2]))
                        +78*(int(pad[i+3,j+3]))-78*(int(pad[i+1,j+1]))
                        +32*(int(pad[i,j+1]))-32*(int(pad[i+4,j+3]))
                        -100*(int(pad[i,j])+int(pad[i+1,j])+int(pad[i+2,j])+int(pad[i+3,j])+int(pad[i+4,j])
                         +int(pad[i+2,j+1])+int(pad[i+3,j+1])+int(pad[i+4,j+1])+int(pad[i+4,j+2]))
                        )
            elif l == 5:
                N.append(100*(int(pad[i,j])+int(pad[i,j+1])+int(pad[i,j+2])+int(pad[i,j+3])+int(pad[i,j+4])
                         +int(pad[i+1,j+2])+int(pad[i+1,j+3])+int(pad[i+1,j+4])+int(pad[i+1,j+4]))
                        +92*(int(pad[i+2,j+3]))-92*(int(pad[i+2,j+1]))
                        +78*(int(pad[i+1,j+1]))-78*(int(pad[i+3,j+3]))
                        +32*(int(pad[i+3,j+4]))-32*(int(pad[i+3,j]))
                        -100*(int(pad[i+2,j])+int(pad[i+3,j])+int(pad[i+3,j+1])+int(pad[i+3,j+2])+int(pad[i+4,j])
                         +int(pad[i+4,j+1])+int(pad[i+4,j+2])+int(pad[i+4,j+3])+int(pad[i+4,j+4]))
                        )
        N = np.array(N)
        if max(N) >= threshold:
            answer[i,j] = 0
        else:
            answer[i,j] = 255
return answer
```

Part(g): do padding first, then do Nevatea_Babu5x5_Operator for every pixel.