(a) dilation

```
def dilation(img, kernel):
    dilation = np.zeros(img.shape)
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            if img[i][j][0] > 0:
                max = 0
                for element in kernel:
                    p,q = element
                    if(i+p >= 0)and(i+p < img.shape[0])and(j+q >= 0)and(j+q <
img.shape[1]):
                        if img[i+p][j+q][0] > max:
                            max = img[i+p][j+q][0]
                for element in kernel:
                    p,q = element
                    if(i+p >= 0)and(i+p < img.shape[0])and(j+q >= 0)and(j+q <
img.shape[1]):
                        dilation[i+p][j+q] = max
    return dilation
cv2.imwrite("dilation.bmp", dilation(img, kernel))
```

- 根據dilation的定義，對於value不為0的pixel，我們要在以該pixel為中心使用kernel可以包含到的範圍中最大的gray level value。使用迴圈的方式掃過整個kernel的範圍，如果kernel覆蓋的位置在圖片中，檢查該位置的gray level value是否比目前的max value大；如果較大則將max value更新成該位置的gray level value，否則維持原數值。

  在計算出max value後，同樣使用迴圈的方式將kernel覆蓋的有效範圍的gray level value更新成max value。

(b) erosion

```
def erosion(img, kernel):
    erosion = np.zeros(img.shape)
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            min = 256
            exist = True
            for element in kernel:
                p,q = element
                if(i+p >= 0)and(i+p < img.shape[0])and(j+q >= 0)and(j+q <
img.shape[1]):
                    if img[i+p][j+q][0] == 0:
                        exist = False
                        break
                    if img[i+p][j+q][0] < min:
                        min = img[i+p][j+q][0]
            exist = True
            for element in kernel:
                p,q = element
                if(i+p >= 0)and(i+p < img.shape[0])and(j+q >= 0)and(j+q <
img.shape[1]):
                    if img[i+p][j+q][0] == 0:
                        exist = False
                        break
                if(i+p >= 0)and(i+p < img.shape[0])and(j+q >= 0)and(j+q <
img.shape[1])and(exist):
                    erosion[i+p][j+q] = min
    return erosion
cv2.imwrite("erosion.bmp", erosion(img, kernel))
```

- erosion的做法和dilation的做法很類似，但要計算的是kernel可覆蓋區域中的不為0的min value，同時須滿足kernel覆蓋的範圍皆需要為非0之數值。在計算出min value後，同樣使用迴圈的方式將kernel覆蓋的有效範圍的gray level value更新成min value。

(c) opening

```
def opening(img, kernel):
    return dilation(erosion(img, kernel),kernel)
cv2.imwrite("opening.bmp", opening(img, kernel))
```

- 根據opening的定義，是先對圖片進行erosion，再對圖片進行dilation。

  使用在(a)(b)已完成的dilation和erosion函數進行組合完成opening的函數操作。

(d) closing

```
def closing(img, kernel):
    return erosion(dilation(img, kernel), kernel)
cv2.imwrite("closing.bmp", closing(img, kernel))
```

- 和opening剛好相反，closing是先對圖片進行dilation，再對圖片進行erosion。

  同樣使用在(a)(b)已完成的dilation和erosion函數進行組合完成closing的函數操作。