Hw4 B11705033 江睿宸

(a) Dilation

遍歷每個 pixel，確定如果加上 structure element 在範圍內，就在新的圖上標成 1，也就是白色

In [3]:
```
structure = [[-2, -1], [-2, 0], [-2, 1],
[-1, -2], [-1, -1], [-1, 0], [-1, 1], [-1, 2],
[0, -2],  [0, -1], [0, 0], [0, 1], [0, 2],
[1, -2],  [1, -1], [1, 0], [1, 1], [1, 2],
          [2, -1], [2, 0], [2, 1]]
```

In [4]:
```
def delation(lena_arr, structure):
    lena_delation = np.zeros_like(lena_arr, dtype=np.uint8)
    for i in range(lena_arr.shape[0]):
        for j in range(lena_arr.shape[1]):
            if lena_arr[i][j] == 1:
                lena_delation[i][j] = 1
                for element in structure:
                    new_i = i + element[0]
                    new_j = j + element[1]
                    if 0 <= new_i < lena_arr.shape[0] and 0 <= new_j < lena_arr.shape[1]:
                        lena_delation[new_i][new_j] = 1
    return lena_delation
```

(b) Erosion

遍歷每個 pixel，如果 structure element 範圍內有一格值為 0，將 temp 旗子標記，只要旗子沒被標記，erosion 後該格就在新的圖上標成 1。

In [12]:
```
def erosion(lena_arr, structure):
    lena_erosion = np.zeros_like(lena_arr, dtype=np.uint8)
    for i in range(lena_arr.shape[0]):
        for j in range(lena_arr.shape[1]):
            lena_erosion[i][j] = 1
            temp = 1
            for element in structure:
                new_i = i + element[0]
                new_j = j + element[1]
                if 0 <= new_i < lena_arr.shape[0] and 0 <= new_j < lena_arr.shape[1]:
                    if lena_arr[new_i][new_j] == 0:
                        temp = 0
                        break
            if temp == 0:
                lena_erosion[i][j] = 0

    return lena_erosion
```

(c) Closing

先 dilation 後 erosion

```
In [16]: lena_closing = erosion(dilation(lena_arr, structure), structure)
         img.fromarray(np.array(lena_closing,dtype='uint8')*255)
```

(d) Opening

先 Erosion 後 Dilation

```
In [12]: lena_opening = dilation(erosion(lena_arr, structure), structure)
         img.fromarray(np.array(lena_opening,dtype='uint8')*255)
```

(e) Hit and miss

標記 structure element，hit and miss = 原 array 對 J 的 erosion 以及其補集對 K 的 erosion，後取兩者交集。

In [14]:
```
struct_j = [[0, -1], [0, 0], [-1, 0]]
struct_k = [[1, 0], [1, 1], [0, 1]]
```

In [15]:
```
lena_j = erosion(lena_arr, struct_j)
lena_com = np.ones_like(lena_arr, dtype=np.uint8)
lena_com = lena_com - lena_arr
lena_k = erosion(lena_com, struct_k)
```

In [16]:
```
hit_and_miss = (lena_j + lena_k)//2
img.fromarray(np.array(hit_and_miss,dtype='uint8')*255)
```