# HW 4

b05902019 資工三 蔡青邑

`Using "lena.bmp" as input image.`

## Python Packages I used

- `skimage.io`: for basic image i/o.
- `numpy`: for convience of array manipulation.
- `tqdm`: for showing the progress of the executing of the code.

## Some Other Functions I Build

- **blank_image(height, width)**: returning an all-black image of the given input height and width.
- **kernel_value(x, y)**: return the kernel value of position (x, y), which, in our case, are always 0.

# Dilation, Erosion, Opening, and Closing

> I wrote a function for each of them, below is how I implement and the image it create.

## *Dilation*

- my function: `dilation(img, kernel)`
- $(f \oplus k)(x, y) = max\{f(x - i, y - j) + k(i, j) | (i, j) \in K, (x - i, y - j) \in f\}$

Following the above equation, I just traverse all the pixels in `img`, finding the local maximum with the kernel applied on each pixel, and that local maximum is the value of the corresponding pixels on `dilation.png`.

# *Erosion*

- my function: `erosion(img, kernel)`
- $(f \ominus k)(x, y) = \min\{f(x + i, y + j) - k(i, j) | (i, j) \in K, (x + i, y + j) \in f\}$

Following the above equation, I just traverse all the pixels in `img`, finding the local minimum with the kernel applied on each pixel, and that local minimum is the value of the corresponding pixels on `erosion.png`.

# *Opening*

- my function: `opening(img, kernel)`
- $B \circ K = (B \ominus K) \oplus K$

Simply apply the formula:

```
def opening(img, kernel):
    temp = erosion(img, kernel)
    temp = dilation(temp, kernel)
    return temp
```

# *Closing*

- my function: `closing(img, kernel)`
- $B \bullet K = (B \oplus K) \ominus K$

Simply apply the formula:

```
def closing(img, kernel):
    temp = dilation(img, kernel)
    temp = erosion(temp, kernel)
    return temp
```