The python script available in “112012049_hm1.py” defines several functions that perform different image processing operations, including converting an image to grayscale, applying convolution operations, applying edge detection convolution, performing max pooling, and binarizing an image.

The script also includes a function that calculates the optimal threshold value using Otsu's method. This function is used to determine the threshold value for binarizing the image.

Finally, the script includes a function that processes the input image through the entire pipeline of image processing operations and displays the resulting image.

Overall, this script provides a useful set of image processing functions that can be used to enhance and analyze images.

Now, let’s explain each of these functions:

- convert_to_grayscale(img_array): This function takes an image array as input and converts it to grayscale. It checks if the image has an alpha channel (transparency) and removes it if it does.
- convolution_operation(image_array, kernel): This function performs a convolution operation on the input image array using the specified kernel. Please note that the kernel chosen for the edge detection is *Sobel Kernel.* Then, a zero padding operation is applied

```
output = np.zeros_like(image_array)

    for i in range(i_height):
        for j in range(i_width):
            region = padded_image[i:i+k_height, j:j+k_width]
            output[i, j] = np.sum(region * kernel)
    return output
```

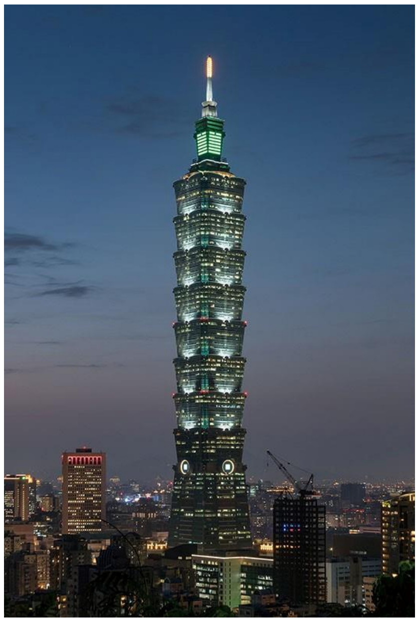
*** Zero padding using a for loop*

- apply_edge_detection_convolution(image_array): This function applies edge detection to the input image array using a convolution operation with a *Sobel kernel*.
- max_pooling(image_array, kernel_size=2, stride=2): This function performs max pooling on the input image array using the specified kernel size and stride.
- binarization (image_array, threshold=128): This function converts the input image array to a binary image using the specified threshold value.
- otsu_threshold(image_array): This function calculates the optimal threshold value for binarization using Otsu's method.

*** Otsu's method is a thresholding technique used to separate an image into foreground and background by finding an optimal threshold value that minimizes the variance within each class and maximizes the variance between the classes.*

- apply_operations(image_path): This function loads an image from the specified file path, processes it through the entire pipeline (grayscale conversion, edge detection, max pooling, and binarization), and returns the resulting binary image as a PIL Image object.

Input image:



Result after each operation: