I. Problem Statement

- Write programs which do binary morphology on a binary image:
  (a) Dilation
  (b) Erosion
  (c) Opening
  (d) Closing
  (e) Hit-and-miss transform

II. Programming Tools

- Programming language: Python 3.8.5
- Library: Numpy 1.19.1, OpenCV 4.0.1

III. Problem-Solving Process

使用 opencv 讀 gray image，再將照片轉為 binary image，threshold 為 128，左圖為本次使用的 3-5-5-5-3 kernel。

```
kernel = np.array([
    [0,1,1,1,0],
    [1,1,1,1,1],
    [1,1,1,1,1],
    [1,1,1,1,1],
    [0,1,1,1,0] ])*255
```

a. Dilation

先對 binary image 做 padding 得到 img_padding，使得 img_padding 上下左右長度加二並設 padding value 為 0，接著判斷 binary imgae 的每個 pixel value，若為 255，則設 kernel 內周遭的 pixel value 為 255。

```
if choice == 0: #  Dilation
    img_padding = np.pad(img,((2,2),(2,2)),'constant',constant_values = (0,0))
    ret = img_padding.copy()
    row,col = img.shape
    for r in range(row):
        for c in range(col):
            if img[r,c] != 255:
                continue
            ret[r:r+5,c:c+5] |= img_padding[r:r+5,c:c+5] | kernel
    return ret[2:-2,2:-2]
```

b. Erosion

先對 binary image 做 padding 得到 img_padding，使得 img_padding 上下左右長度加二並設 padding value 為 0，接著判斷 binary imgae 每個 pixel value 在 kernel 範圍內是否都為 255，若皆為 255，則設定該 pixel value 為 255，否則設為 0。

```
elif choice == 1: # Erosion
    img_padding = np.pad(img,((2,2),(2,2)),'constant',constant_values = (0,0))
    ret = np.zeros_like(img)
    row,col = img.shape
    kernel = kernel.astype(np.bool)
    for r in range(row):
        for c in range(col):
            if img_padding[r:r+5,c:c+5][kernel].all():
                ret[r,c] = 255
    return ret
```

c. Opening

先對 binary image 做 erosion，在對 erosion 過後的 image 做 dilation。

```
elif choice == 2: # Opening
    return solve(solve(img,1,kernel),0,kernel)
```

d. Closing

先對 binary image 做 dilation，在對 erosion 過後的 image 做 erosion。

```
elif choice == 3: # Closing
    return solve(solve(img,0,kernel),1,kernel)
```

e. Hit-and-miss transform

使用下圖的 j_kernel 對 binary image 做 erosion，再用下圖的 k_kernel 對 inverse 過後的 binary image 做 erosion，最後對兩張 erosion 後的 image 做 and operation 得到 Hit-and-miss transform 後的 image。

```
elif choice == 4: # Hit-and-miss transform
    j_kernel = np.array([
        [0,0,0,0,0],
        [0,0,0,0,0],
        [1,1,0,0,0],
        [0,1,0,0,0],
        [0,0,0,0,0] ])*255
    k_kernel = np.array([
        [0,0,0,0,0],
        [0,1,1,0,0],
        [0,0,1,0,0],
        [0,0,0,0,0],
        [0,0,0,0,0] ])*255
    reverse = -img + 255
    return solve(img,1,j_kernel) & solve(reverse,1,k_kernel)
```

IV. Results

a. Dilation

b. Erosion

c. Opening

d. Closing

e. Hit-and-miss transform