(a) Dilation on gray-scale image

Algorithm:

$$(f \oplus b)(x) = \sup_{y \in E}[f(y) + b(x - y)]$$

Code fragment:

```
kernel = [[-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]]
```

```
def dilation(img, kernel):
    r = img.shape[0]
    c = img.shape[1]
    new_img = np.zeros(img.shape, dtype=int)
    for i in range(r):
        for j in range(c):
            localMax = 0
            for d in kernel:
                dr, dc = d
                if (i+dr) >= 0 and (i+dr) < r and (j+dc) >= 0 and (j+dc) < c:
                    localMax = max(img[i+dr][j+dc], localMax)
            new_img[i][j] = localMax
    return new_img
```

Result:

(b) Erosion on gray-scale image

Algorithm:

$$(f \ominus b)(x) = \inf_{y \in B}[f(x + y) - b(y)]$$

Code fragment:

```python
def erosion(img, kernel):
    r = img.shape[0]
    c = img.shape[1]
    new_img = np.zeros(img.shape, dtype=np.uint8)
    for i in range(r):
        for j in range(c):
            localMin = 255
            for d in kernel:
                dr, dc = d
                if (i+dr) >= 0 and (i+dr) < r and (j+dc) >= 0 and (j+dc) < c:
                    localMin = min(img[i+dr][j+dc], localMin)
            new_img[i][j] = localMin
    return new_img
```

Result:

(c) Opening on gray-scale image

Algorithm:

$$A \circ B = (A \ominus B) \oplus B$$

Code fragment:

```python
def open(img, kernel):
    return dilation(erosion(img, kernel), kernel)
```

Result:

(d) Closing on gray-scale image

Algorithm:

$$A \bullet B = (A \oplus B) \ominus B$$

Code fragment:

```
def close(img, kernel):
    return erosion(dilation(img, kernel), kernel)
```

Result: