```
import numpy as np
conv_kernel = np.array([[-1, 0, 1],
                        [-1, 0, 1],
                        [-1, 0, 1]])

data_matrix = np.array([
    [2, 2, 2, 2, 2, 1, 1, 1],
    [2, 2, 2, 2, 2, 1, 1, 1],
    [2, 2, 2, 2, 2, 1, 1, 1],
    [2, 2, 2, 2, 2, 1, 1, 1],
    [2, 2, 2, 9, 9, 9, 9, 9],
    [2, 2, 2, 9, 9, 9, 9, 9],
    [2, 2, 2, 9, 9, 9, 9, 9],
    [2, 2, 2, 9, 9, 9, 9, 9]
])



def Conv2(data_matrix, conv_kernel, stride, bias, padding, paddin
    if padding:
        img2d = np.pad(data_matrix,padding_h, mode='constant',
                       constant_values=0)
    else:
        img2d = data_matrix
    inw, inh = img2d.shape
    w, h = conv_kernel.shape
    outwidth = (inw - w) // stride + 1
    outheight = (inh - h) // stride + 1
    arrayy = np.zeros(shape=(outwidth, outheight))
    for i in range(outheight):
        for j in range(outwidth):
            s = 0
            for k in range(w):
                for l in range(h):
                    s += img2d[k + i * stride][l + j * stride] *
            arrayy[i][j] = s + bias
    return arrayy
```

```
def MaxPooling(data_matrix, pool_kernel, stride, padding, padding
    if padding:
        img2d = np.pad(data_matrix,padding_h, mode='constant',
                       constant_values=0)
    else:
        img2d = data_matrix
    inw, inh = img2d.shape
    w, h = pool_kernel, pool_kernel
    outwidth = (inw - w) // stride + 1
    outheight = (inh - h) // stride + 1
    arrayy_max = np.zeros(shape=(outwidth, outheight))
    for i in range(outheight):
        for j in range(outwidth):
            s = []
            for k in range(w):
                for l in range(h):
                    small_s = img2d[k + i * stride][l + j * strid
                    s.append(small_s)
            arrayy_max[i][j] = np.array(s).max()
    return arrayy_max


def AvgPooling(data_matrix, pool_kernel, stride, padding, padding
    if padding:
        img2d = np.pad(data_matrix,padding_h, mode='constant',
                       constant_values=0)
    else:
        img2d = data_matrix
    inw, inh = img2d.shape
    w, h = pool_kernel, pool_kernel
    outwidth = (inw - w) // stride + 1
    outheight = (inh - h) // stride + 1
    arrayy_avg = np.zeros(shape=(outwidth, outheight))
    for i in range(outheight):
        for j in range(outwidth):
            s = []
            for k in range(w):
                for l in range(h):
                    small_s = img2d[k + i * stride][l + j * strid
```

```
                    s.append(small_s)
            arrayy_avg[i][j] = round(np.array(s).mean(), 4)
    return arrayy_avg
```

Answer for question 3

```
conv_value = Conv2(data_matrix=data_matrix, conv_kernel=conv_kern
                   padding='True', padding_h=1)
print(conv_value)
```

```
[[  4.5   6.5   6.5   6.5   5.5   4.5   3.5   2.5]
 [  0.5   0.5   0.5   0.5   0.5   0.5   0.5   0.5]
 [  0.5   0.5   0.5   0.5   0.5   0.5   0.5   0.5]
 [  0.5   0.5   7.5  14.5  22.5  23.5  24.5  16.5]
 [  0.5   0.5   7.5  14.5  22.5  23.5  24.5  16.5]
 [  0.5   0.5   0.5   0.5   0.5   0.5   0.5   0.5]
 [  0.5   0.5   0.5   0.5   0.5   0.5   0.5   0.5]
 [ -3.5  -5.5 -12.5 -19.5 -26.5 -26.5 -26.5 -17.5]]
```

Answer for question 4

```
conv_value_stride2 = Conv2(data_matrix=data_matrix, conv_kernel=c
                           padding='padding', padding_h=1)
print(conv_value_stride2)
```

```
[[ 4.5  6.5  5.5  3.5]
 [ 0.5  0.5  0.5  0.5]
 [ 0.5  7.5 22.5 24.5]
 [ 0.5  0.5  0.5  0.5]]
```

Answer for question 5(a)

```
max_pool_ans = MaxPooling(data_matrix=conv_value, pool_kernel=3,
print(max_pool_ans)
```

```
[[ 6.5  6.5  6.5  6.5  6.5  5.5  4.5  3.5]
 [ 6.5  6.5  6.5  6.5  6.5  5.5  4.5  3.5]
 [ 0.5  7.5 14.5 22.5 23.5 24.5 24.5 24.5]
 [ 0.5  7.5 14.5 22.5 23.5 24.5 24.5 24.5]
 [ 0.5  7.5 14.5 22.5 23.5 24.5 24.5 24.5]
```

```
 [ 0.5  7.5 14.5 22.5 23.5 24.5 24.5 24.5]
 [ 0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5]
 [ 0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5]]
```

Answer for question 5(b)

```
avg_pool_ans = AvgPooling(data_matrix=conv_value, pool_kernel=3,
print(avg_pool_ans)
```

```
[[ 1.3333  2.1111  2.3333  2.2222  2.      1.6667  1.3333  0.7778]
 [ 1.4444  2.2778  2.5     2.3889  2.1667  1.8333  1.5     0.8889]
 [ 0.3333  1.2778  2.8333  5.2778  7.0556  8.1667  7.5     4.7778]
 [ 0.3333  2.0556  5.1667 10.0556 13.6111 15.8333 14.5     9.2222]
 [ 0.3333  2.0556  5.1667 10.0556 13.6111 15.8333 14.5     9.2222]
 [ 0.3333  1.2778  2.8333  5.2778  7.0556  8.1667  7.5     4.7778]
 [-0.7778 -2.0556 -3.8333 -6.1667 -7.7222 -8.5    -7.5    -4.6667]
 [-0.8889 -2.2222 -4.     -6.3333 -7.8889 -8.6667 -7.6667 -4.7778]]
```

✓ 0s completed at 3:53 PM