Exercise 3 - (E7.1)

The required modules:

In [1]:
```python
import numpy as np

from utilities.activation_functions import hardlims
```

Point 1

Let's define the problem variables. We then convert the squares into vectors, by scanning them column by column. White squares are represented as -1, conversely blue ones as 1:

In [2]:
```python
p1 = np.array([-1, -1, 1, 1]).reshape(-1, 1)
p2 = np.array([1, 1, -1, 1]).reshape(-1, 1)
```

The orthogonality can be easily checked by recalling that two vectors are orthogonal if:

$$\mathbf{p}_1^T \cdot \mathbf{p}_2 = 0 \tag{1}$$

In [3]:
```python
# Orthogonality check
print("Dot product: ", np.squeeze(np.dot(p1.transpose(), p2)))
```

```
Dot product:  -2
```

The dot product is not zero, therefore **the two vectors are not orthogonal**.

Point 2

Using the Hebb rule in the case of **autoassociative network**, thus the output to each input being the input itself, the weight matrix is computed as:

$$\mathbf{W} = \mathbf{p}_1\mathbf{p}_1^T + \mathbf{p}_2\mathbf{p}_2^T = \mathbf{P}\mathbf{P}^T \tag{2}$$

In [4]:
```python
# Weight matrix
W = np.dot(p1, p1.transpose()) + np.dot(p2, p2.transpose())
print("Hebb W matrix:\n", W)
```

```
Hebb W matrix:
 [[ 2  2 -2  0]
 [ 2  2 -2  0]
 [-2 -2  2  0]
 [ 0  0  0  2]]
```

Point 3

Let us define the test input pattern $\mathbf{p}_t$:

In [5]:
```python
pt = np.array([1, 1, 1, 1]).reshape(-1, 1)
```

Recovering the implementation of the predict function in the previous exercises, excluding the bias:

In [6]:
```python
# Layer output function
def predict(p: np.ndarray, W_fun: np.ndarray):
    return hardlims(np.dot(W_fun, p.reshape(-1, 1)))
```

The prediction can be carried out as:

In [7]:
```python
y_pred = predict(pt, W)

print("Prediction: ", np.squeeze(y_pred))
```

```
Prediction:  [ 1  1 -1  1]
```

The prediction is **coincidentally equal** to the input $\mathbf{p}_2$, in fact the simple Hebb rule cannot assure the output will correctly match the respective input if those are not orthogonal. When the prototype input patterns are not orthogonal, the Hebb rule produces some errors, and the use of the pseudoinverse rule can reduce errors related to non-orthogonality.