In [1]:

```
import matplotlib.pyplot as plt
import numpy as np
```

In [2]:

```
np.random.seed(1000)

nb_patterns = 4
pattern_width = 4
pattern_height = 4
max_iterations = 10
```

In [3]:

```
X = np.zeros((nb_patterns, pattern_width * pattern_height))
print(X)
```

```
[[ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]]
```

In [11]:

```
X[0] = [-1, 1, 1, -1, -1, 1, 1, -1, -1, 1, 1, -1, -1, 1, 1, -1]
X[1] = [-1, -1, -1, -1, 1, 1, 1, 1, 1, 1, 1, 1, -1, -1, -1, -1]
X[2] = [-1, -1, 1, 1, -1, -1, 1, 1, 1, 1, -1, -1, 1, 1, -1, -1]
X[3] = [1, 1, -1, -1, 1, 1, -1, -1, -1, -1, 1, 1, -1, -1, 1, 1]
print(X[0])
```

```
[-1.  1.  1. -1. -1.  1.  1. -1. -1.  1.  1. -1. -1.  1.  1. -1.]
```

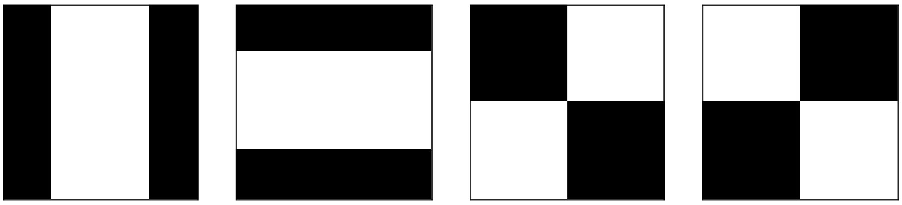
In [5]:

```
fig, ax = plt.subplots(1, nb_patterns, figsize=(10, 5))
```

In [6]:

```
for i in range(nb_patterns):
    ax[i].matshow(X[i].reshape((pattern_height, pattern_width)), cmap='gray')
    ax[i].set_xticks([])
    ax[i].set_yticks([])

plt.show()
```



In [15]:

```
W = np.zeros((pattern_width * pattern_height, pattern_width * pattern_height))
print(W)
print(X.shape[0])
```

```
[[ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]]
4
```

In [16]:

```
for i in range(pattern_width * pattern_height):
    for j in range(pattern_width * pattern_height):
        w=0.0
        if i!=j and W[i,j]==0:
            for l in range(0,nb_patterns):
                w=w+X[l,i]*X[l,j]
            W[i,j]=w/X.shape[0] #normalize
            W[j,i]=W[i,j] #diagonal same value
print(W)
```

```
[[ 0.   0.5 -0.5  0.   0.5  0.  -1.  -0.5 -0.5 -1.   0.   0.5  0.  -0.5
   0.5  1. ]
 [ 0.5  0.   0.  -0.5  0.   0.5 -0.5 -1.  -1.  -0.5  0.5  0.  -0.5  0.   1.
   0.5]
 [-0.5  0.   0.   0.5 -1.  -0.5  0.5  0.   0.   0.5 -0.5 -1.   0.5  1.   0.
  -0.5]
 [ 0.  -0.5  0.5  0.  -0.5 -1.   0.   0.5  0.5  0.  -1.  -0.5  1.   0.5
  -0.5  0. ]
 [ 0.5  0.  -1.  -0.5  0.   0.5 -0.5  0.   0.  -0.5  0.5  1.  -0.5 -1.   0.
   0.5]
 [ 0.   0.5 -0.5 -1.   0.5  0.   0.  -0.5 -0.5  0.   1.   0.5 -1.  -0.5
   0.5  0. ]
 [-1.  -0.5  0.5  0.  -0.5  0.   0.   0.5  0.5  1.   0.  -0.5  0.   0.5
  -0.5 -1. ]
 [-0.5 -1.   0.   0.5  0.  -0.5  0.5  0.   1.   0.5 -0.5  0.   0.5  0.  -1.
  -0.5]
 [-0.5 -1.   0.   0.5  0.  -0.5  0.5  1.   0.   0.5 -0.5  0.   0.5  0.  -1.
  -0.5]
 [-1.  -0.5  0.5  0.  -0.5  0.   1.   0.5  0.5  0.   0.  -0.5  0.   0.5
  -0.5 -1. ]
 [ 0.   0.5 -0.5 -1.   0.5  1.   0.  -0.5 -0.5  0.   0.   0.5 -1.  -0.5
   0.5  0. ]
 [ 0.5  0.  -1.  -0.5  1.   0.5 -0.5  0.   0.  -0.5  0.5  0.  -0.5 -1.   0.
   0.5]
 [ 0.  -0.5  0.5  1.  -0.5 -1.   0.   0.5  0.5  0.  -1.  -0.5  0.   0.5
  -0.5  0. ]
 [-0.5  0.   1.   0.5 -1.  -0.5  0.5  0.   0.   0.5 -0.5 -1.   0.5  0.   0.
  -0.5]
 [ 0.5  1.   0.  -0.5  0.   0.5 -0.5 -1.  -1.  -0.5  0.5  0.  -0.5  0.   0.
   0.5]
 [ 1.   0.5 -0.5  0.   0.5  0.  -1.  -0.5 -0.5 -1.   0.   0.5  0.  -0.5
   0.5  0. ]]
```

In [27]:

```
# Create a corrupted test pattern
x_test = np.array([1, -1, 1, 1, -1, -1, 1, 1, -1, 1, -1, -1, 1, 1, 1, 1])
x_test2=np.array([-1, 1, 1, -1, -1, -1, -1, -1, -1, 1, 1, -1, -1, 1, 1, -1])
```

In [20]:

```
# Recover the original patterns
A = x_test.copy()
print(np.dot(W[0],A))
for _ in range(max_iterations):
    for i in range(pattern_width * pattern_height):
        if np.dot(W[i], A) > 0:
            A[i] = 1.0
        else:
            A[i]= -1.0
print(A[0])
print(A)
```

```
-3.0
-1
[-1 -1  1  1 -1 -1  1  1  1  1 -1 -1  1  1 -1 -1]
```

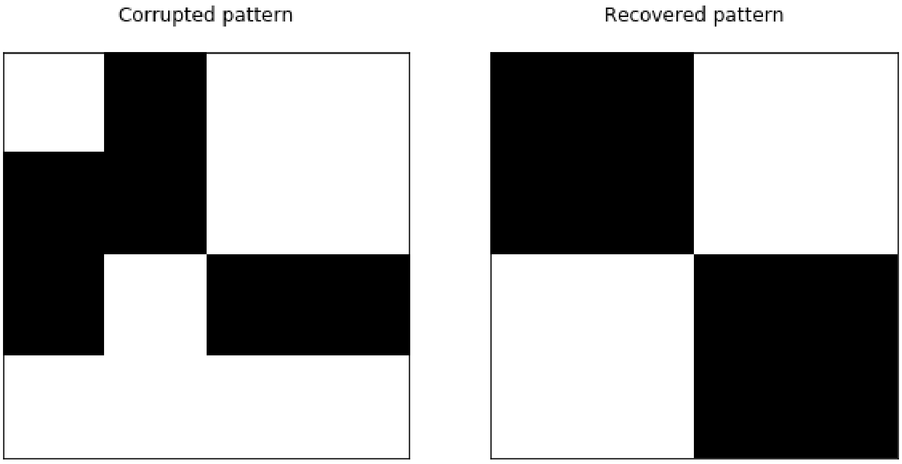
In [19]:

```
fig, ax = plt.subplots(1, 2, figsize=(10, 5))

ax[0].matshow(x_test.reshape(pattern_height, pattern_width), cmap='gray')
ax[0].set_title('Corrupted pattern')
ax[0].set_xticks([])
ax[0].set_yticks([])

ax[1].matshow(A.reshape(pattern_height, pattern_width), cmap='gray')
ax[1].set_title('Recovered pattern')
ax[1].set_xticks([])
ax[1].set_yticks([])

plt.show()
```



In [28]:

```
B = x_test2.copy()
print(np.dot(W[0],B))
for _ in range(max_iterations):
    for i in range(pattern_width * pattern_height):
        if np.dot(W[i], B) > 0:
            B[i] = 1.0
        else:
            B[i]= -1.0
print(B[0])
print(B)
```

```
-1.0
-1
[-1  1  1 -1 -1  1  1 -1 -1  1  1 -1 -1  1  1 -1]
```

In [29]:

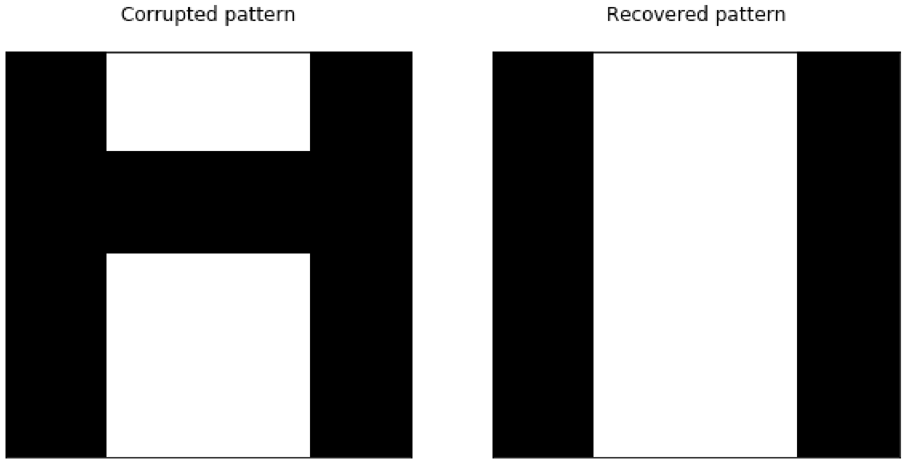
```
fig, ax = plt.subplots(1, 2, figsize=(10, 5))

ax[0].matshow(x_test2.reshape(pattern_height, pattern_width), cmap='gray')
ax[0].set_title('Corrupted pattern')
ax[0].set_xticks([])
ax[0].set_yticks([])

ax[1].matshow(B.reshape(pattern_height, pattern_width), cmap='gray')
```

```
ax[1].set_title('Recovered pattern')
ax[1].set_xticks([])
ax[1].set_yticks([])

plt.show()
```



Corrupted pattern

Recovered pattern

In [ ]: