In [1]:

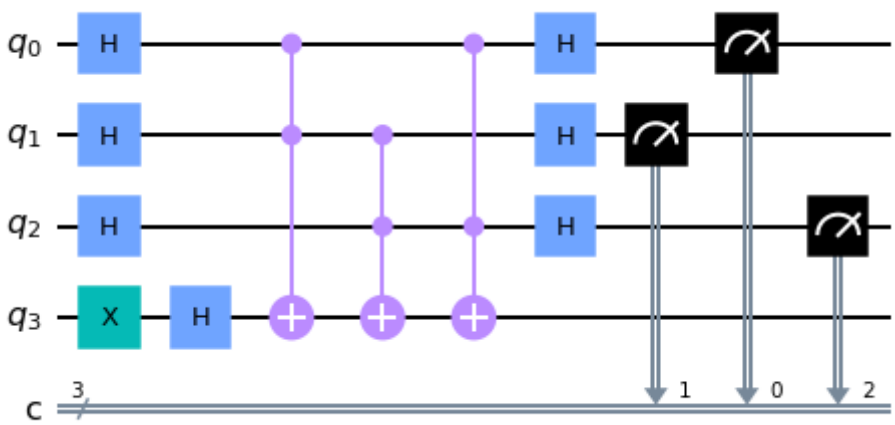
```
import numpy as np

# Importing standard Qiskit libraries
from qiskit import QuantumCircuit, transpile, Aer, IBMQ
from qiskit.tools.jupyter import *
from qiskit.visualization import *
from ibm_quantum_widgets import *
from qiskit.providers.aer import QasmSimulator

# Loading your IBM Quantum account(s)
provider = IBMQ.load_account()
```

In [17]:

```
from qiskit import*
qr=QuantumRegister(4,'q')
cr=ClassicalRegister(3,'c')
qc=QuantumCircuit(qr,cr)
for i in range(3):
    qc.h(qr[i])
qc.x(qr[3])
qc.h(qr[3])
qc.ccx(qr[0],qr[1],qr[3])
qc.ccx(qr[1],qr[2],qr[3])
qc.ccx(qr[0],qr[2],qr[3])
for i in range(3):
    qc.h(qr[i])
for i in range(3):
    qc.measure(qr[i],cr[i])
backend=Aer.get_backend('qasm_simulator')
qjob=execute(qc,backend,shorts=2000)
counts=qjob.result().get_counts()
print(counts)
qc.draw()
```

```
{'100': 257, '111': 240, '010': 257, '001': 270}
```

Out[17]:



In [21]:

```
from qiskit import*
qr=QuantumRegister(2,'q')
cr=ClassicalRegister(1,'c')
qc=QuantumCircuit(qr,cr)
qc.h(qr[0])
qc.x(qr[1])
qc.h(qr[1])
qc.cx(qr[0],qr[1])
qc.x(qr[0])
qc.cx(qr[0],qr[1])
qc.x(qr[0])
qc.h(qr[0])
qc.measure(qr[0],cr[0])
backend=Aer.get_backend('qasm_simulator')
qjob=execute(qc,backend,shorts=500)
counts=qjob.result().get_counts()
print(counts)
qc.draw()
```

```
{'0': 1024}
```

Out[21]: