1. Try the experiments above with different gates (CNOT, Controlled-S, Controlled-T†), what results do you expect? What results do you get?

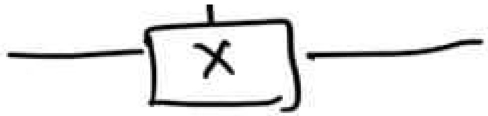
In [1]:

```python
#initialization
import matplotlib.pyplot as plt
import numpy as np
import math

# importing Qiskit
from qiskit import IBMQ, Aer, transpile, assemble
from qiskit import QuantumCircuit, ClassicalRegister, QuantumRegister

# import basic plot tools
from qiskit.visualization import plot_histogram
```

In [2]:

```python
def qft_dagger(qc, n):
    """n-qubit QFTdagger the first n qubits in circ"""
    # Don't forget the Swaps!
    for qubit in range(n//2):
        qc.swap(qubit, n-qubit-1)
    for j in range(n):
        for m in range(j):
            qc.cp(-math.pi/float(2**(j-m)), m, j)
        qc.h(j)
```

1. Phase Estimation on CNOT



CNOT $\rightarrow$ controlled - X

$U \equiv X$ is unitary

$U|u\rangle = e^{2\pi i \theta}|u\rangle$ ——— ⊛

where $|u\rangle$ is the eigenvector of U.

X has eigenvalue $\longrightarrow 1 \longrightarrow$ eigenvector $\begin{pmatrix}1\\1\end{pmatrix}$

$X\begin{pmatrix}1\\1\end{pmatrix} = \begin{pmatrix}0&1\\1&0\end{pmatrix}\begin{pmatrix}1\\1\end{pmatrix} = \begin{pmatrix}1\\1\end{pmatrix}$

$\Rightarrow X\begin{pmatrix}1\\1\end{pmatrix} = 1\begin{pmatrix}1\\1\end{pmatrix}$ compare with ⊛

$\Rightarrow e^{2\pi i\theta} = 1 = e^0 \Rightarrow \boxed{\theta = 0}$

$\longrightarrow -1 \longrightarrow$ eigenvector $\begin{pmatrix}-1\\1\end{pmatrix}$

$X\begin{pmatrix}-1\\1\end{pmatrix} = \begin{pmatrix}0&1\\1&0\end{pmatrix}\begin{pmatrix}-1\\1\end{pmatrix} = \begin{pmatrix}1\\-1\end{pmatrix} = -1\begin{pmatrix}-1\\1\end{pmatrix}$

$$\Rightarrow X\begin{pmatrix}-1\\1\end{pmatrix} = -1\begin{pmatrix}1\\-1\end{pmatrix}$$ compare with ⊛

$$e^{2\pi i\theta} = -1 = e^{\pi i}$$

$$\Rightarrow 2\pi i\theta = \pi i \Rightarrow \boxed{\theta = \frac{1}{2}}$$

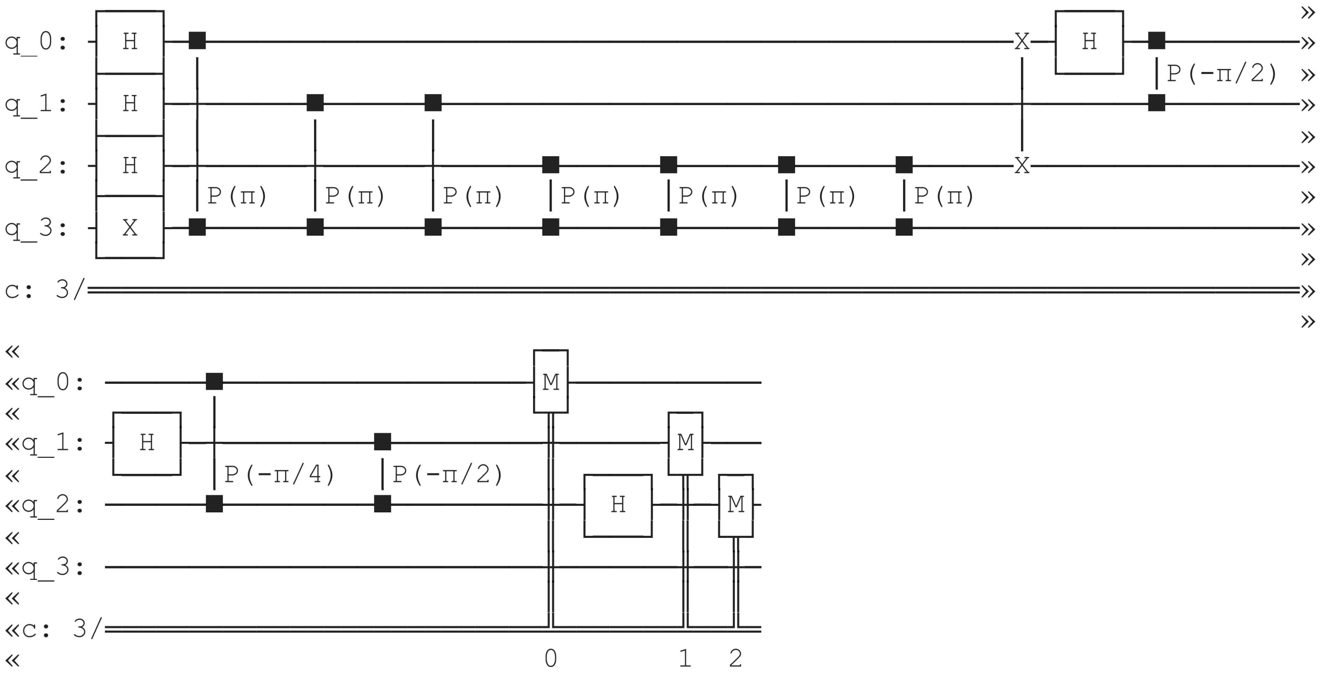
We will use phase as $\theta = \frac{1}{2}$

In [3]:

```python
# Create and set up circuit
qpe2 = QuantumCircuit(4, 3)

# Apply H-Gates to counting qubits:
for qubit in range(3):
    qpe2.h(qubit)

# Prepare our eigenstate |psi>:
qpe2.x(3)

# Do the controlled-U operations:
angle = 2*math.pi/2
repetitions = 1
for counting_qubit in range(3):
    for i in range(repetitions):
        qpe2.cp(angle, counting_qubit, 3);
    repetitions *= 2
```

```python
# Do the inverse QFT:
qft_dagger(qpe2, 3)

# Measure of course!
for n in range(3):
    qpe2.measure(n,n)

qpe2.draw()
```

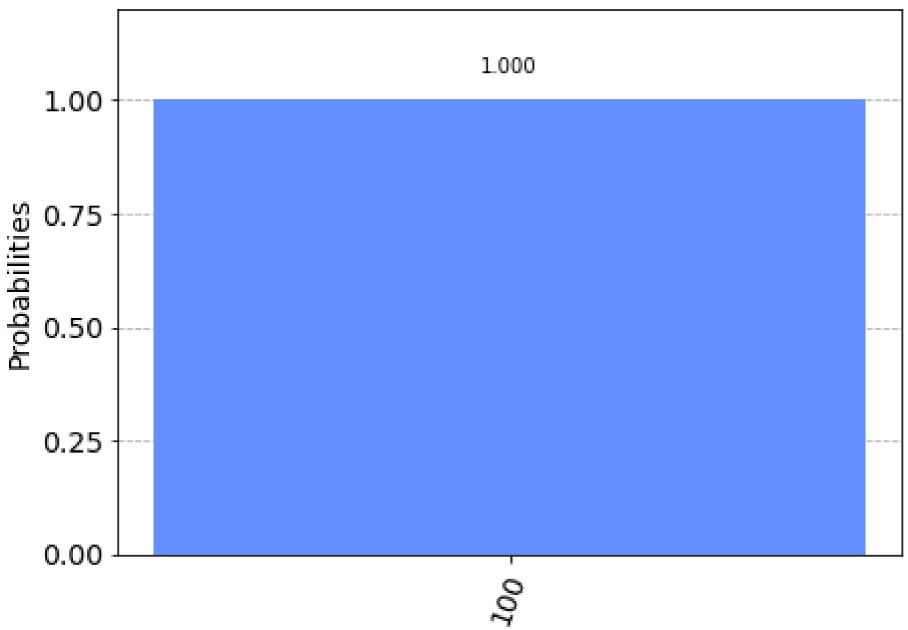
Out[3]:



In [4]:

```python
# Let's see the results!
aer_sim = Aer.get_backend('aer_simulator')
shots = 4096
t_qpe2 = transpile(qpe2, aer_sim)
qobj = assemble(t_qpe2, shots=shots)
results = aer_sim.run(qobj).result()
answer = results.get_counts()

plot_histogram(answer)
```

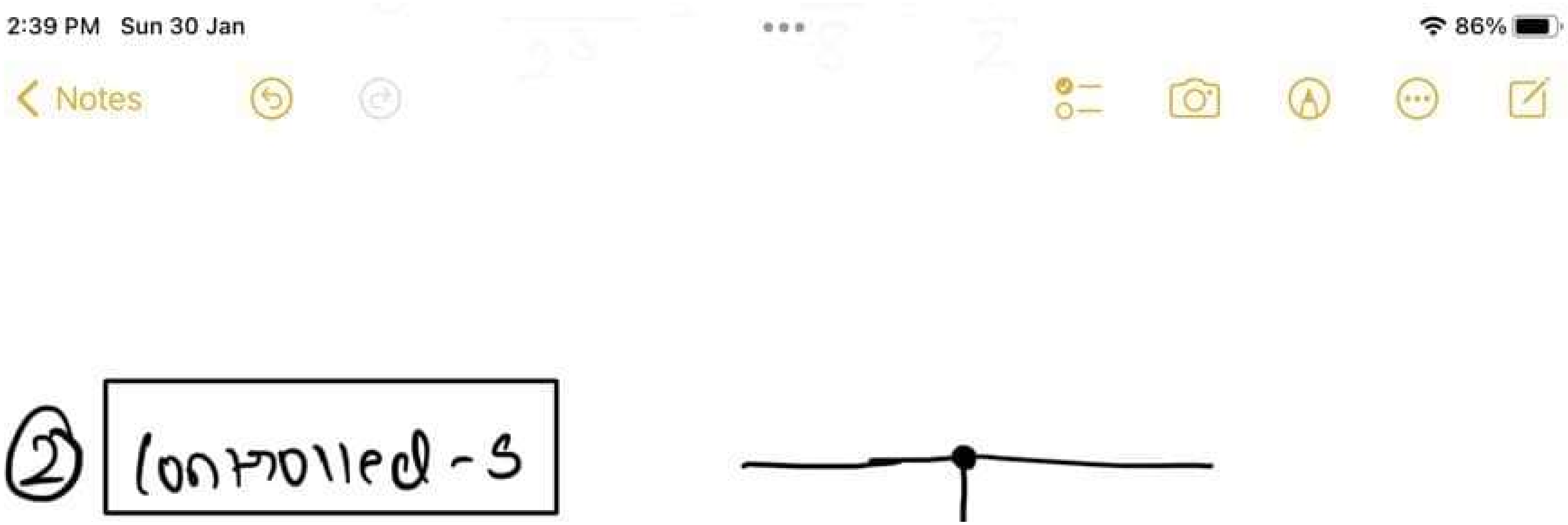
Out[4]:


1.000
1.00
0.75
0.50
0.25
0.00
Probabilities
100

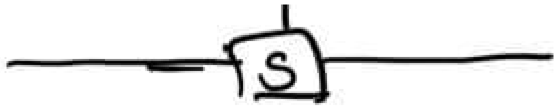
We get the result as 100 which is 4 in decimal.

$$\theta = \frac{4}{2^3} = \frac{4}{8} = \frac{1}{2}$$

2. Phase Estimation on Controlled-S



$U \equiv S$ is unitary



$S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$ S gate adds a phase of $e^{i\frac{\pi}{2}}$ to the state $|1\rangle$

$$\boxed{e^{i\frac{\pi}{2}} = i \quad \left\{ \begin{aligned} e^{i\phi} &= \cos\phi + i\sin\phi \\ e^{i\frac{\pi}{2}} &= \cos\frac{\pi}{2} + i\sin\frac{\pi}{2} \\ &= 0 + i\cdot 1 = i \end{aligned} \right\}}$$

$$S|1\rangle = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ i \end{bmatrix} = \begin{bmatrix} 0 \\ e^{i\pi/2} \end{bmatrix}$$

$$= e^{i\frac{\pi}{2}}|1\rangle$$

i.e $S|1\rangle = e^{i\frac{\pi}{2}}|1\rangle$

Compare with 4

$$e^{2\pi i\theta} = e^{i\pi/2}$$ above

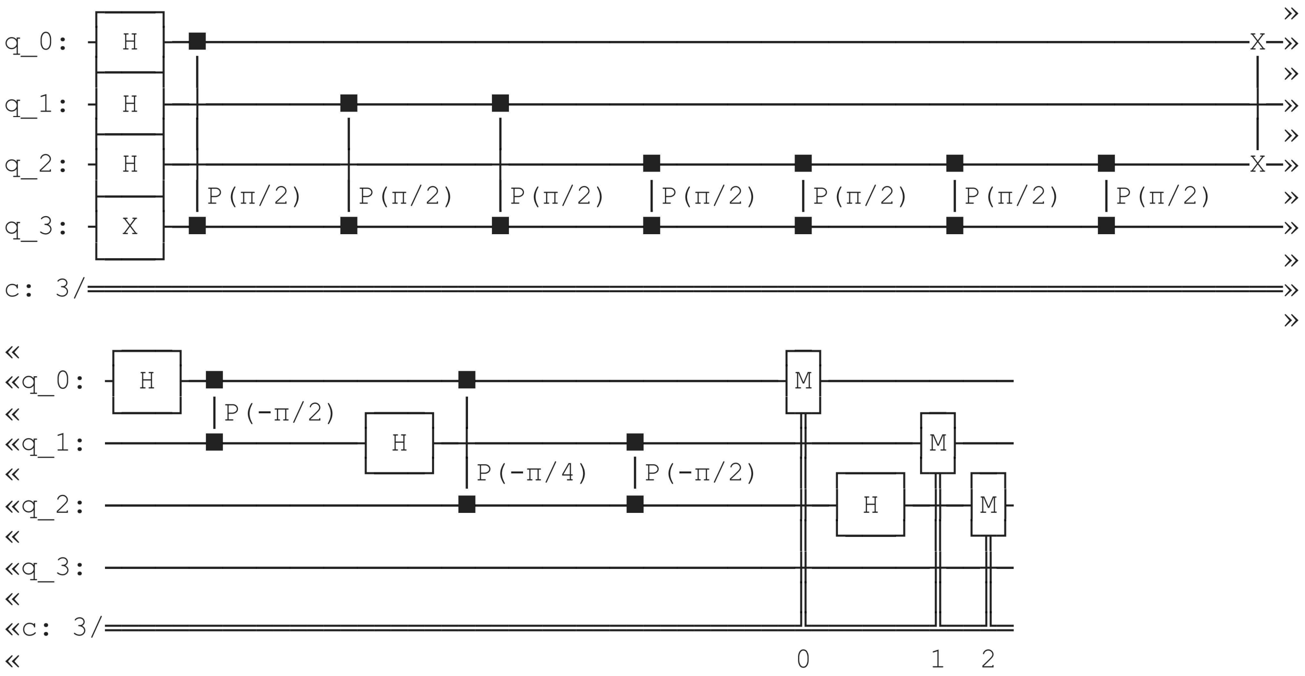
$$\Rightarrow 2\pi i\theta = i\frac{\pi}{2} \Rightarrow \boxed{\theta = \frac{1}{4}}$$

In [5]:

```
# Create and set up circuit
qpe2 = QuantumCircuit(4, 3)

# Apply H-Gates to counting qubits:
for qubit in range(3):
    qpe2.h(qubit)

# Prepare our eigenstate |psi>:
qpe2.x(3)

# Do the controlled-U operations:
angle = 2*math.pi/4
repetitions = 1
for counting_qubit in range(3):
    for i in range(repetitions):
        qpe2.cp(angle, counting_qubit, 3);
    repetitions *= 2

# Do the inverse QFT:
qft_dagger(qpe2, 3)

# Measure of course!
for n in range(3):
    qpe2.measure(n,n)
```

```python
qpe2.draw()
```

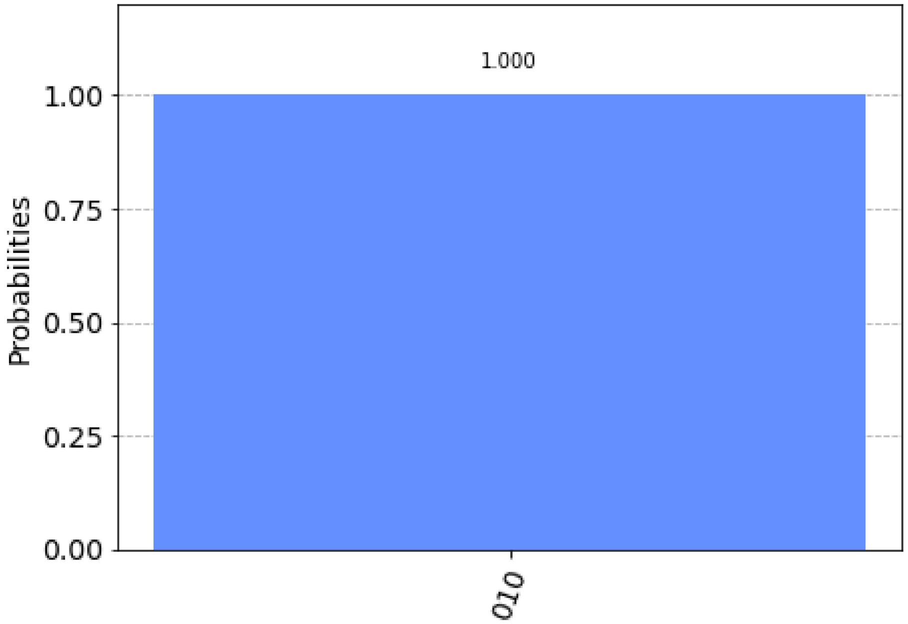
Out[5]:



In [6]:

```python
# Let's see the results!
aer_sim = Aer.get_backend('aer_simulator')
shots = 4096
t_qpe2 = transpile(qpe2, aer_sim)
qobj = assemble(t_qpe2, shots=shots)
results = aer_sim.run(qobj).result()
answer = results.get_counts()

plot_histogram(answer)
```

Out[6]:



1.000
1.00
0.75
0.50
0.25
0.00
Probabilities
010

we get the output as 010 which in decimal is 2

$$\theta = \frac{2}{2^3} = \frac{2}{8} = \frac{1}{4}$$

3. Phase Estimation on Controlled-T+

2:40 PM Sun 30 Jan 86%

Notes

(3) $\boxed{\text{Controlled - } T^+}$

$$T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix} \qquad T^{+} = \begin{pmatrix} 0 & e^{-i\pi/4} \end{pmatrix}$$

$$-\frac{\pi}{4} \Rightarrow 2\pi - \frac{\pi}{4} = \frac{7\pi}{4}$$

$$\Rightarrow T^{+} = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\,7\pi/4} \end{pmatrix}$$

{ we can also take as $e^{-i\pi/4}$ }

$$\Rightarrow T^{+}|1\rangle = e^{i\,7\pi/4}\,|1\rangle$$

Compare with (*)

$$e^{2\pi i\theta} = e^{i\,7\pi/4}$$

$$\Rightarrow 2\pi i\theta = i\,7\frac{\pi}{4} \Rightarrow \boxed{\theta = \frac{7}{8}}$$

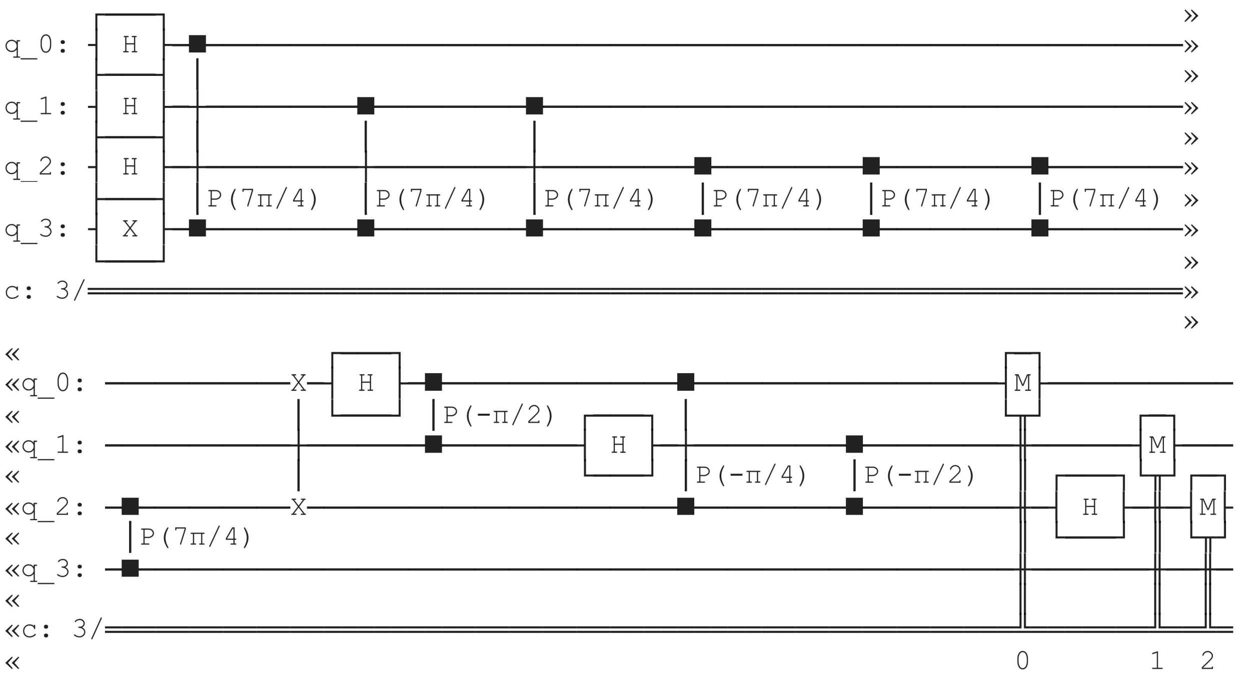
or $\boxed{\theta = -\frac{1}{8}}$

In [7]:

```
# Create and set up circuit
qpe2 = QuantumCircuit(4, 3)

# Apply H-Gates to counting qubits:
```

```
for qubit in range(3):
    qpe2.h(qubit)

# Prepare our eigenstate |psi>:
qpe2.x(3)

# Do the controlled-U operations:
angle = 7*2*math.pi/8
repetitions = 1
for counting_qubit in range(3):
    for i in range(repetitions):
        qpe2.cp(angle, counting_qubit, 3);
    repetitions *= 2

# Do the inverse QFT:
qft_dagger(qpe2, 3)

# Measure of course!
for n in range(3):
    qpe2.measure(n,n)

qpe2.draw()
```

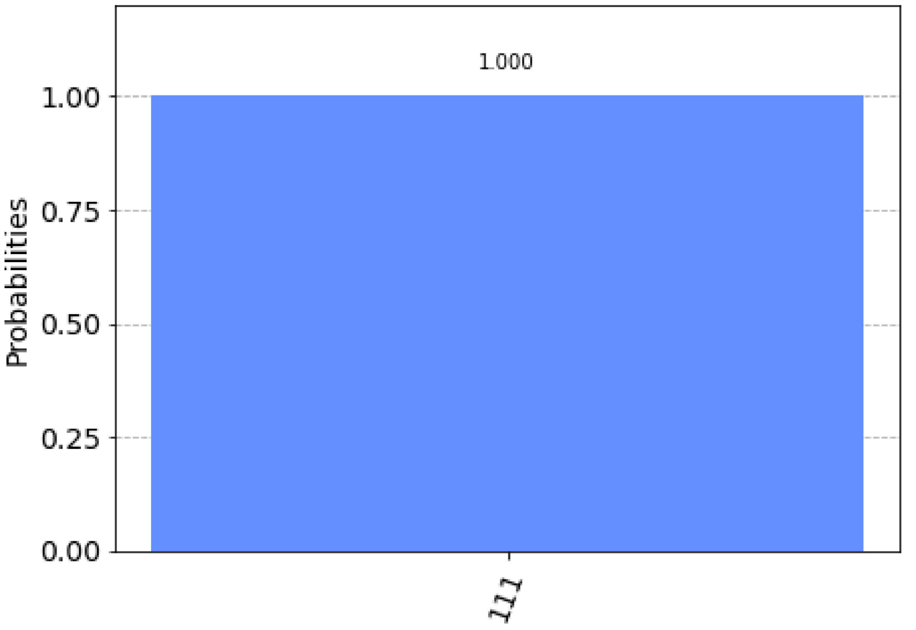
Out[7]:



In [8]:

```
# Let's see the results!
aer_sim = Aer.get_backend('aer_simulator')
shots = 4096
t_qpe2 = transpile(qpe2, aer_sim)
qobj = assemble(t_qpe2, shots=shots)
results = aer_sim.run(qobj).result()
answer = results.get_counts()

plot_histogram(answer)
```

Out[8]:



we get the output as 111, which in decimal is 7

$$\theta = \frac{7}{2^3} = \frac{7}{8}$$

we can $\frac{7}{8}$, even if we use $\theta = -\frac{1}{8}$

2. Try the experiment with a Controlled- Y-gate, do you get the result you expected?

(ii) $\boxed{\text{Controlled} - Y}$

$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \longrightarrow$ Eigenvalue $1 \rightarrow$ eigenvector $\begin{pmatrix} -i \\ 1 \end{pmatrix}$

$$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}\begin{pmatrix} -i \\ 1 \end{pmatrix} = 1\begin{pmatrix} -i \\ 1 \end{pmatrix}$$

$Y\begin{pmatrix} -i \\ 1 \end{pmatrix} = 1\begin{pmatrix} -i \\ 1 \end{pmatrix}$ Compare with $*$

$$e^{2\pi i\theta} = e^{0}$$

$$\Rightarrow 2\pi i\theta = 0 \Rightarrow \boxed{\theta = 0}$$

$\longrightarrow$ Eigenvalue $-1 \rightarrow$ Eigenvector $\begin{pmatrix} i \\ 1 \end{pmatrix}$

$$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}\begin{pmatrix} i \\ 1 \end{pmatrix} = \begin{pmatrix} -i \\ -1 \end{pmatrix} = -1\begin{pmatrix} i \\ 1 \end{pmatrix}$$

$Y\begin{pmatrix} i \\ 1 \end{pmatrix} = -1\begin{pmatrix} i \\ 1 \end{pmatrix}$ Compare with $*$

$$e^{2\pi i\theta} = -1 = e^{\pi i}$$

$\Rightarrow 2\pi i\theta = \pi i$ $\Rightarrow \boxed{\theta = \frac{1}{2}}$

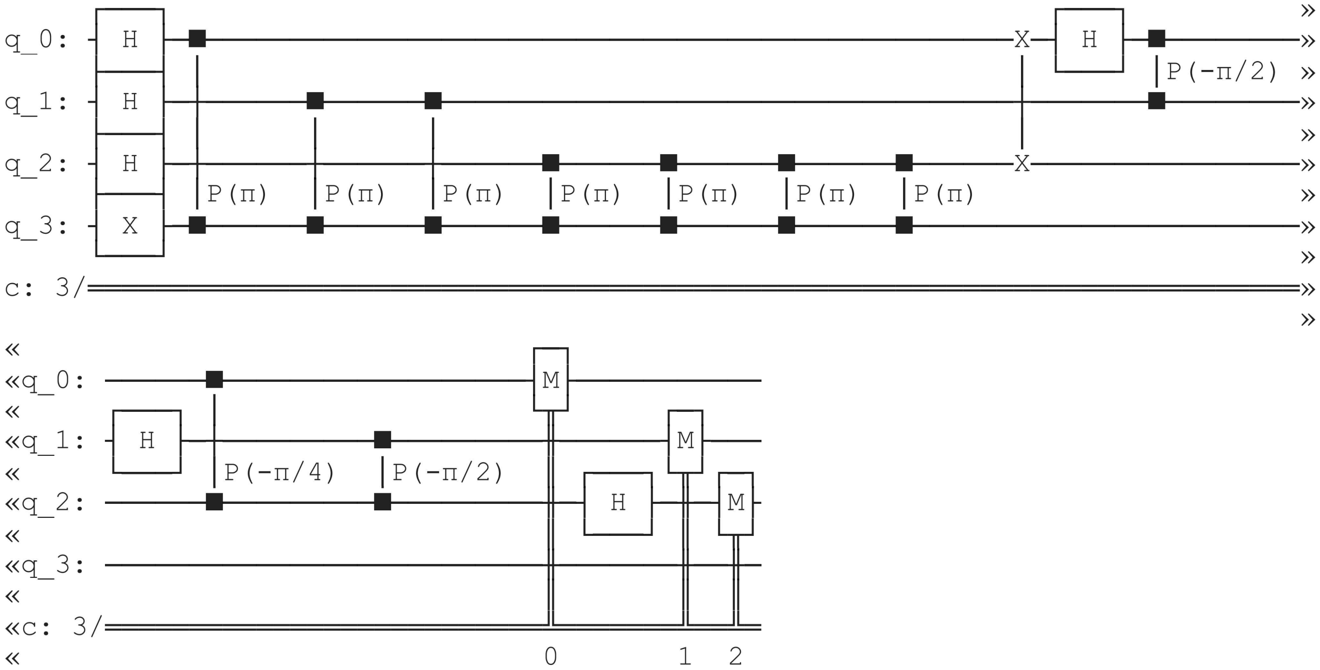
We will use phase as $\theta = \frac{1}{2}$.

In [9]:

```
# Create and set up circuit
qpe2 = QuantumCircuit(4, 3)

# Apply H-Gates to counting qubits:
for qubit in range(3):
    qpe2.h(qubit)

# Prepare our eigenstate |psi>:
qpe2.x(3)

# Do the controlled-U operations:
angle = 2*math.pi/2
repetitions = 1
for counting_qubit in range(3):
    for i in range(repetitions):
        qpe2.cp(angle, counting_qubit, 3);
    repetitions *= 2
```

```python
# Do the inverse QFT:
qft_dagger(qpe2, 3)

# Measure of course!
for n in range(3):
    qpe2.measure(n,n)

qpe2.draw()
```

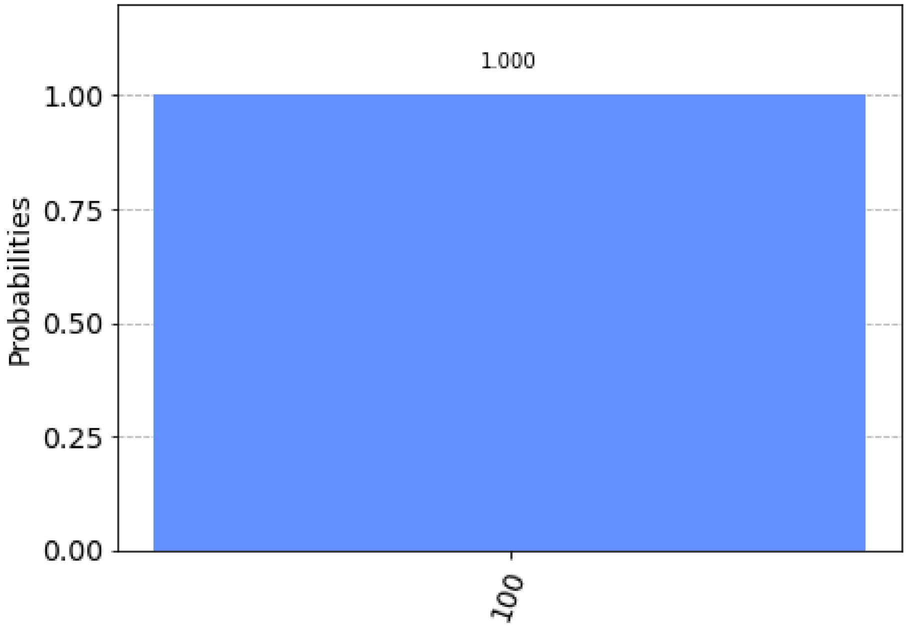
Out[9]:



In [10]:

```python
# Let's see the results!
aer_sim = Aer.get_backend('aer_simulator')
shots = 4096
t_qpe2 = transpile(qpe2, aer_sim)
qobj = assemble(t_qpe2, shots=shots)
results = aer_sim.run(qobj).result()
answer = results.get_counts()

plot_histogram(answer)
```

Out[10]:


1.000
1.00
0.75
0.50
0.25
0.00
Probabilities
100

We get the result as 100 which is 4 in decimal.

$$\theta = \frac{4}{2^3} = \frac{4}{8} = \frac{1}{2}$$

In [ ]: