In [1]:

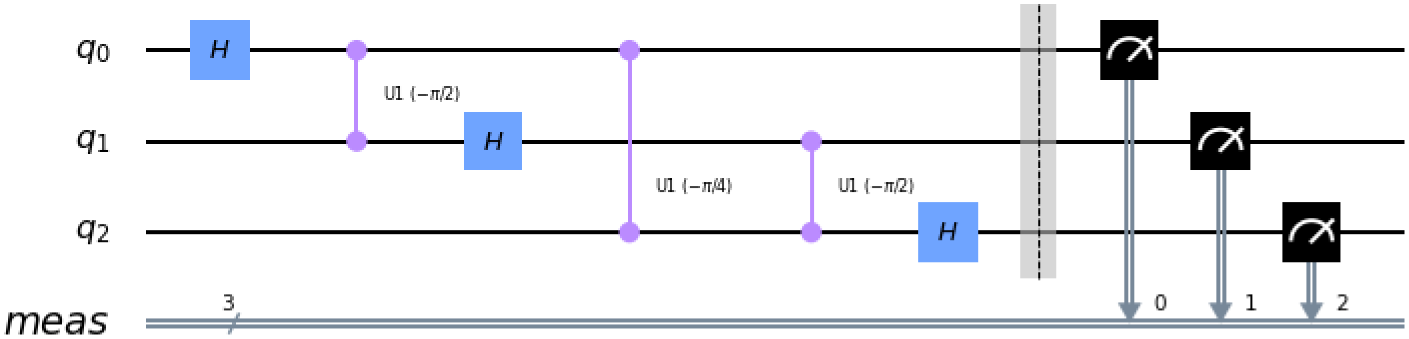
```
import numpy as np
from numpy import pi
from qiskit import *
from qiskit.visualization import plot_histogram
%matplotlib inline
```

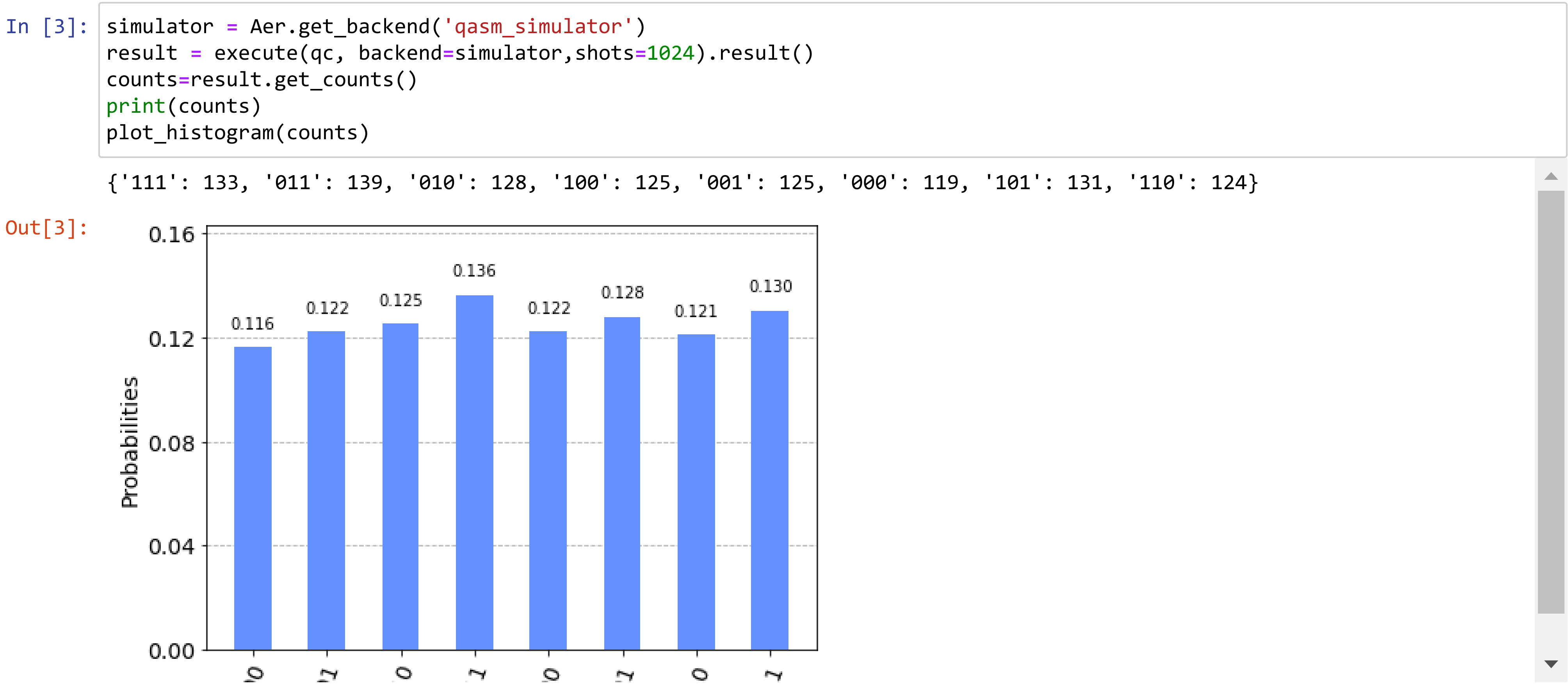
In [2]:

```
#Applying QFT
n=int(input())
qc=QuantumCircuit(n-1)
for j in range(n-1):
    for m in range(j):
        qc.cu1(-pi/float(2**(j-m)), m, j)
    qc.h(j)

qc.measure_all()
qc.draw(output='mpl')
```

```
4
```

Out[2]:





In [3]:

```
simulator = Aer.get_backend('qasm_simulator')
result = execute(qc, backend=simulator,shots=1024).result()
counts=result.get_counts()
print(counts)
plot_histogram(counts)
```

```
{'111': 133, '011': 139, '010': 128, '100': 125, '001': 125, '000': 119, '101': 131, '110': 124}
```

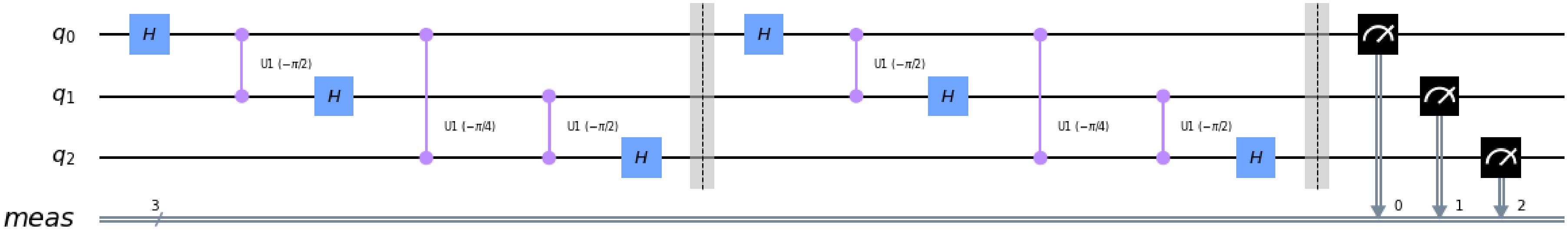
Out[3]:

In [4]:

```python
#applying QFT
n=int(input())
qc=QuantumCircuit(n-1)
for j in range(n-1):
    for m in range(j):
        qc.cu1(-pi/float(2**(j-m)), m, j)
    qc.h(j)

qc.barrier()

#Applying Inverse QFT
for j in range(n-1):
    for m in range(j):
        qc.cu1(-pi/float(2**(j-m)), m, j)
    qc.h(j)

qc.measure_all()
qc.draw(output='mpl')
```

```
4
```

Out[4]:

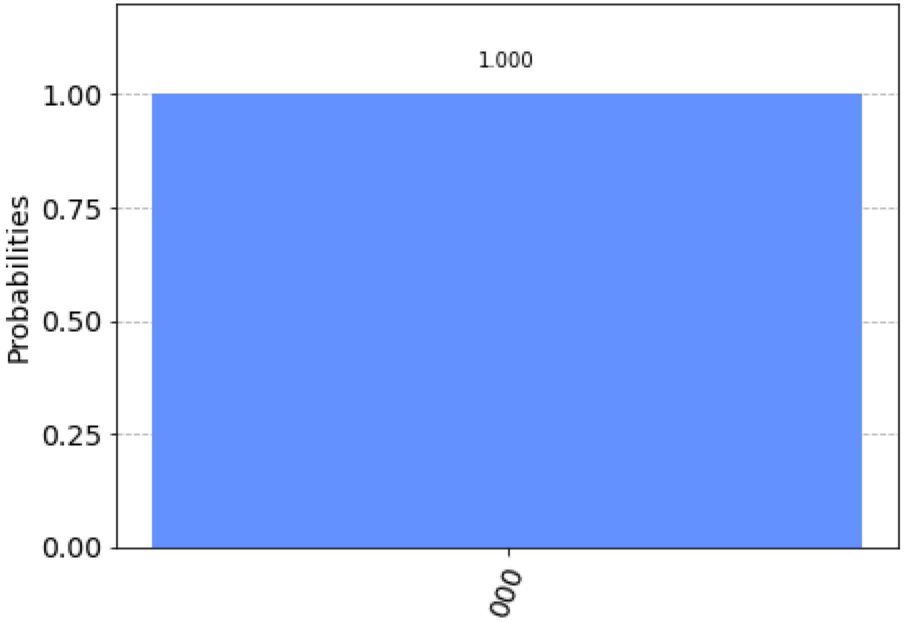


In [5]:

```
simulator = Aer.get_backend('qasm_simulator')
result = execute(qc, backend=simulator,shots=1024).result()
counts=result.get_counts()
print(counts)
plot_histogram(counts)
```

```
{'000': 1024}
```

Out[5]:



In [ ]: