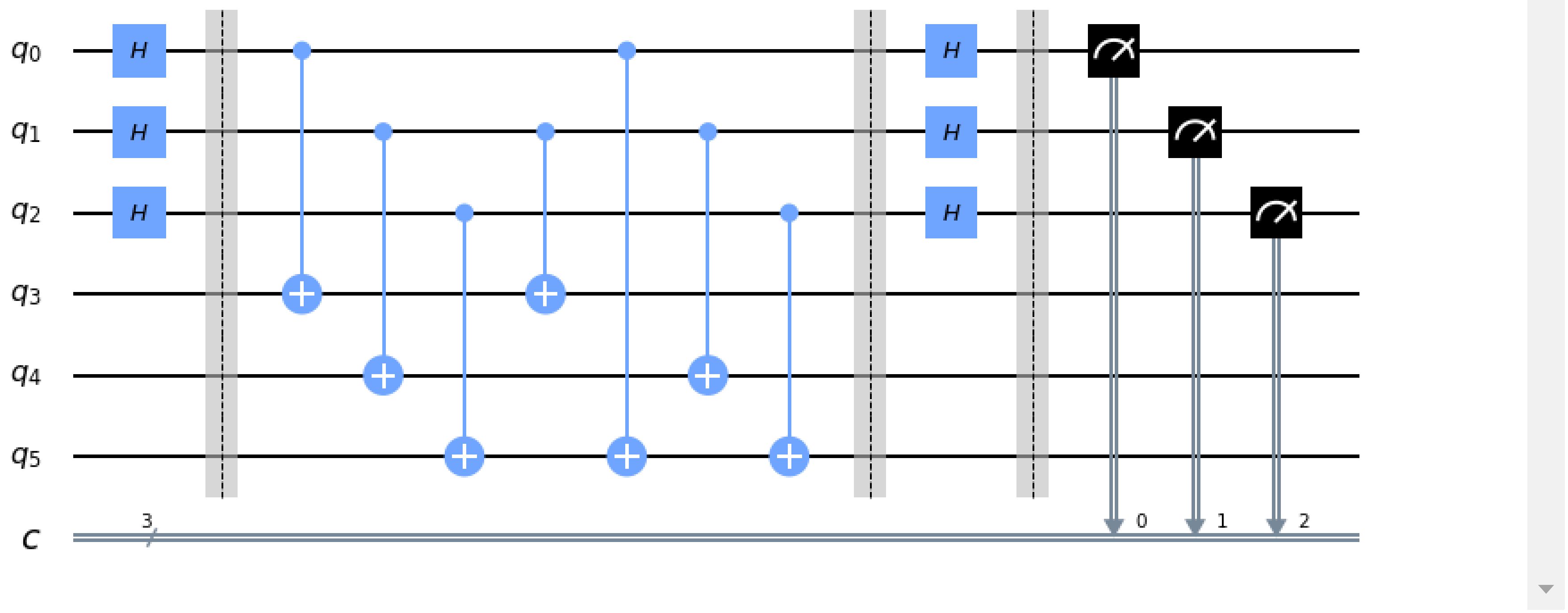
In [1]:

```
from qiskit import *
import matplotlib.pyplot as plt
import numpy as np
from qiskit.visualization import plot_histogram
```

In [2]:

```
%matplotlib inline
qr=QuantumRegister(6)
cr=ClassicalRegister(3)
sc=QuantumCircuit(6,3)
sc.h([0])
sc.h([1])
sc.h([2])

sc.barrier()

#blackbox for function defined as
#f(000)=000
#f(001)=000
#f(010)=100
#f(011)=100
#f(100)=110
#f(101)=110
#f(110)=010
#f(111)=010

sc.cx([0],[3])
sc.cx([1],[4])
sc.cx([2],[5])
sc.cx([1],[3])
sc.cx([1],[4])
sc.cx([0],[5])
sc.cx([2],[5])

sc.barrier()

sc.h([0])
sc.h([1])
sc.h([2])

sc.barrier()

sc.measure([0],[0])
sc.measure([1],[1])
sc.measure([2],[2])

sc.draw(output='mpl')
```

Out[2]:

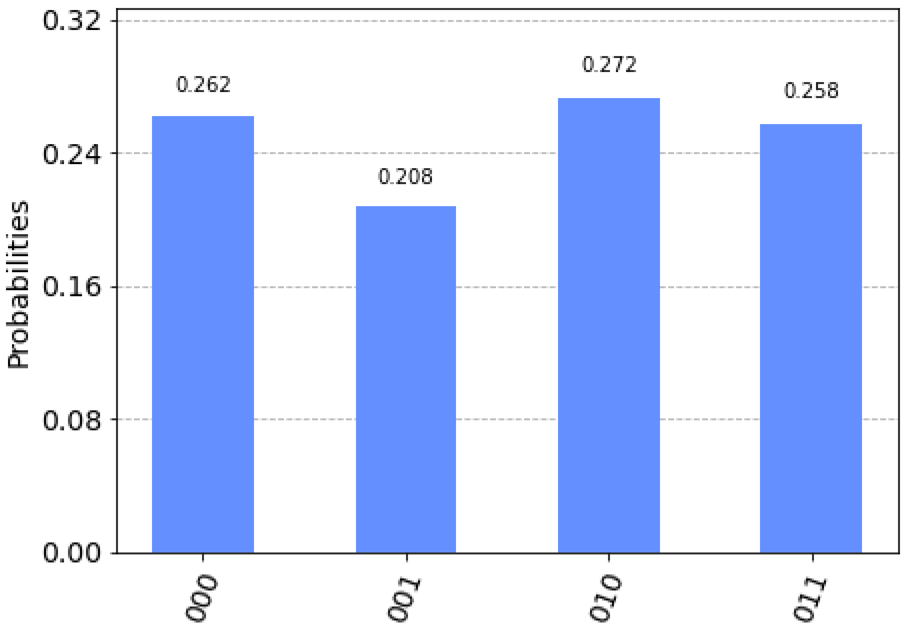

q_0
q_1
q_2
q_3
q_4
q_5
c
3
H
H
H
H
H
H
0
1
2

In [3]:

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

```
{'011': 264, '000': 268, '010': 279, '001': 213}
```

Out[3]:



In [4]:

```
b='100'
def bdotz(b, z):
    accum = 0
    for i in range(len(b)):
        accum += int(b[i]) * int(z[i])
    return (accum % 2)

for z in counts:
    print( '{}.{} = {} (mod 2)'.format(b, z, bdotz(b,z)) )
```

```
100.011 = 0 (mod 2)
100.000 = 0 (mod 2)
100.010 = 0 (mod 2)
100.001 = 0 (mod 2)
```

In [ ]: