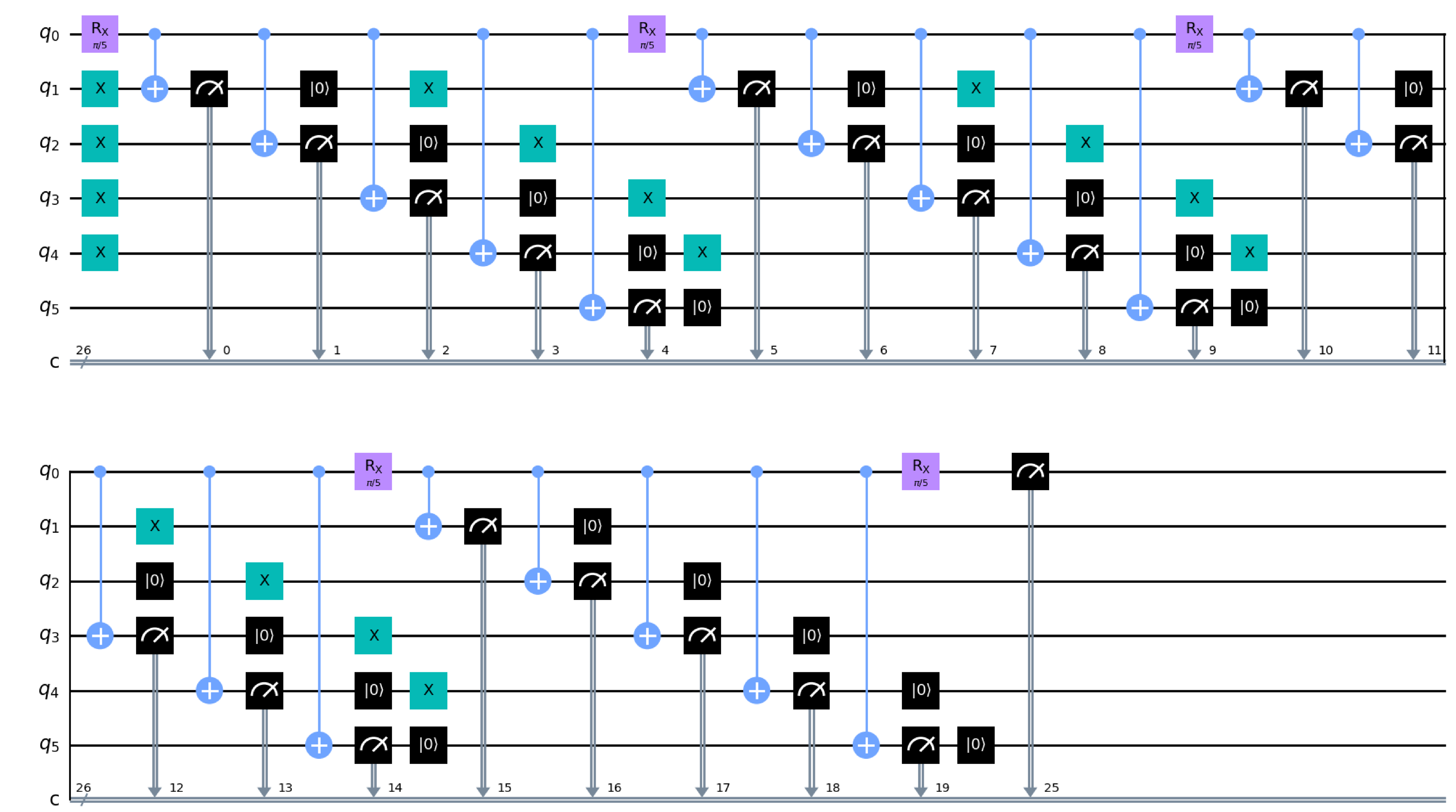
In [78]:

```
#Tadeh Khajatourians
#CS3650 Final Project
#---------------------------------------------------------------#
#This project is a fun little game where a quantum ghost buster  #
#is trying to find a ghost, while trying to not spook it away.   #
#This game was inspired by the "Quantum Minesweeper" game.       #
#---------------------------------------------------------------#
from qiskit import QuantumCircuit, QuantumRegister, ClassicalRegister
from qiskit.visualization import plot_histogram
from qiskit.circuit.library import RXGate
from random import randint
import numpy as np

#How many times we repeat the search cycle is determined below
#The more times we repeat the cycle it means more runtime, but higher probability of not spooking the ghost while getting the correct answer
cycles = 5
#Theta represents how much we are going to rotate the beam splitter for each measure
theta = np.pi/cycles

#Randomly place the ghost in the so called maze
ghost_index = randint(1,5)

def quantumGhostBuster(cycles) -> QuantumCircuit:
    #One for each possible place the ghost can be found
    #One quantuam ghost buster
    qr = QuantumRegister(6, 'q')
    cr = ClassicalRegister(cycles*5+1, 'c') #cycles*5+1 to keep in range
    qc = QuantumCircuit(qr, cr)
    for cycle in range(cycles-1):
        qc.append(RXGate(theta), [qr[0]])
        for ghost in range(1,6):
            if (ghost!=ghost_index):
                qc.x(ghost) #If there is no ghost found in this path, then we apply a NOT gate
            qc.cx(qr[0], qr[ghost]) #The quantum ghost buster is using a CNOT gate as a way of checking for the ghost
            qc.measure(qr[ghost],cr[cycle*5+ghost-1])
            if cycle < cycles-1:
                qc.reset(qr[ghost])
    #Final gate check
    qc.append(RXGate(theta), [qr[0]])
    qc.measure(qr[0],cr[cycles*5])
    return qc

successes = 0 # Track the number of successful predictions
zeno_circuit = quantumGhostBuster(cycles)
zeno_circuit.draw(output='mpl')
```

Out[78]:



In [112…

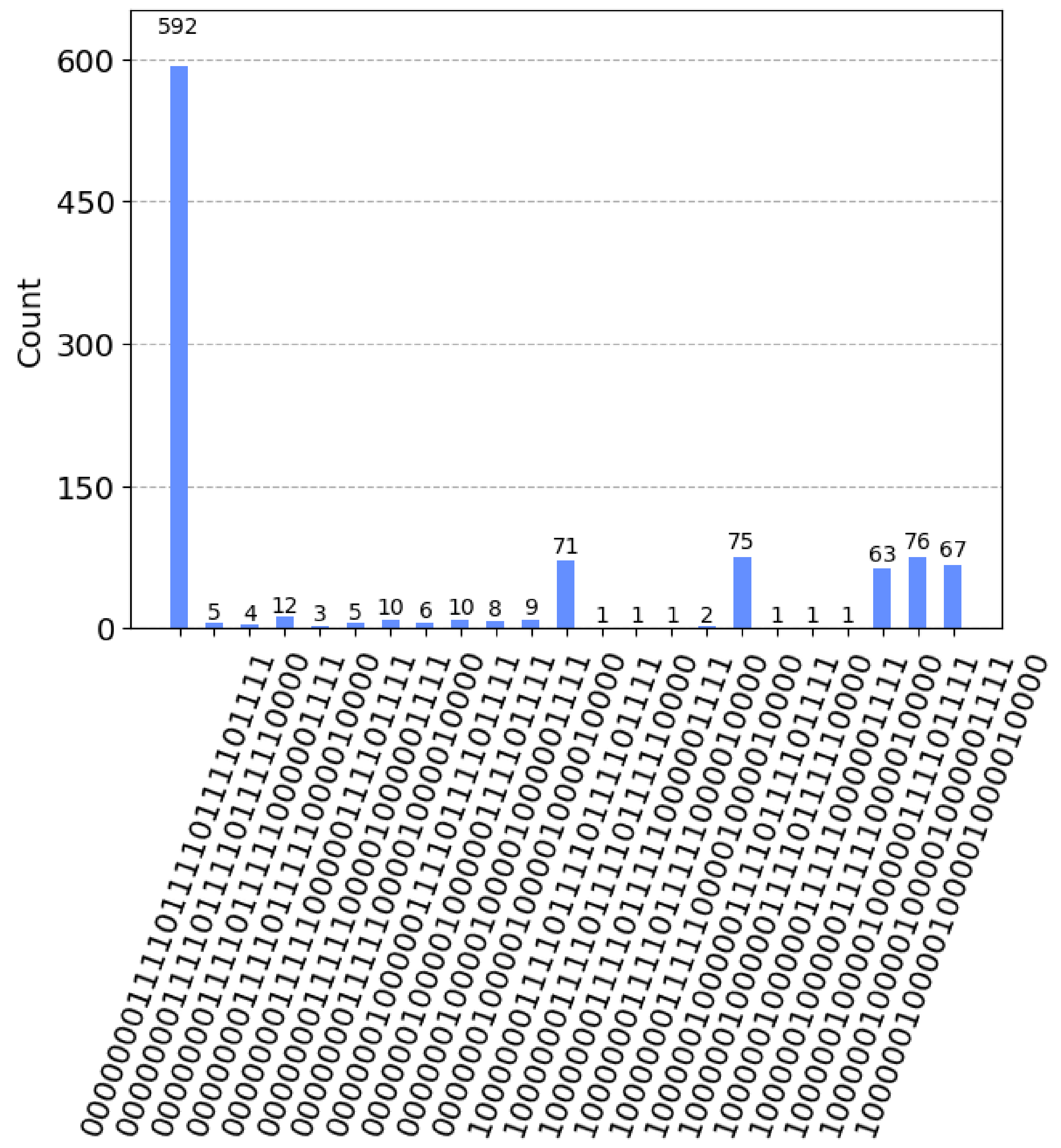
```
from qiskit.providers.aer import QasmSimulator
simulator = QasmSimulator()
#Running the zeno tester
qsweeper_job = simulator.run(zeno_circuit, shots=1500)
qsweeper_result = qsweeper_job.result()
qsweeper_counts = qsweeper_result.get_counts(zeno_circuit)

#The indices that we are checking for a correct measurement,
#Which is where the ghost buster finds the ghost AND does not spook it away
check_indices = []
max_index = cycles*5+1
for cycle in range(cycles-1):
    # registers read back to front the way qiskit is set up
    index = max_index-(ghost_index*(cycle*1+1))
    check_indices.append(index)
check_indices.append(0)


correct_count = 0
for possibility in qsweeper_counts:
    correct = True
    for index in check_indices:
        if (possibility[index]!='0'):
            correct = False

    if (correct==True):
        correct_count = qsweeper_counts[possibility]

print("The ghost buster spotted the ghost " + str(correct_count/10) + "% of the time while not spooking the it away!")
```

```
The ghost buster spotted the ghost 89.3% of the time while not spooking the it away!
```

In [58]:

```
from qiskit.visualization import plot_histogram
plot_histogram(qsweeper_counts)
```

Out[58]:



In [59]:

```
print(qsweeper_counts)
```

```
{'10000001111011110111101111': 71, '10000010000100000111101111': 63, '10000010000100001000001111': 76, '00000010000011110111101111': 6, '00000010000100000111101111': 10, '00000001111011110111101111': 59
2, '00000001111100001000001111': 5, '10000010000011110111101111': 75, '00000001111100000111101111': 3, '10000010000100001000010000': 67, '00000010000100001000010000': 9, '00000010000100001000001111': 8,
'00000001111100001000010000': 10, '00000001111011111000010000': 12, '00000001111011111000001111': 4, '10000001111100001000010000': 2, '00000001111011110111110000': 5, '10000001111011111000001111': 1, '1
0000010000011110111110000': 1, '10000001111011111000010000': 1, '10000001111011110111110000': 1, '10000010000011111000001111': 1, '10000010000011111000010000': 1}
```

In [ ]: