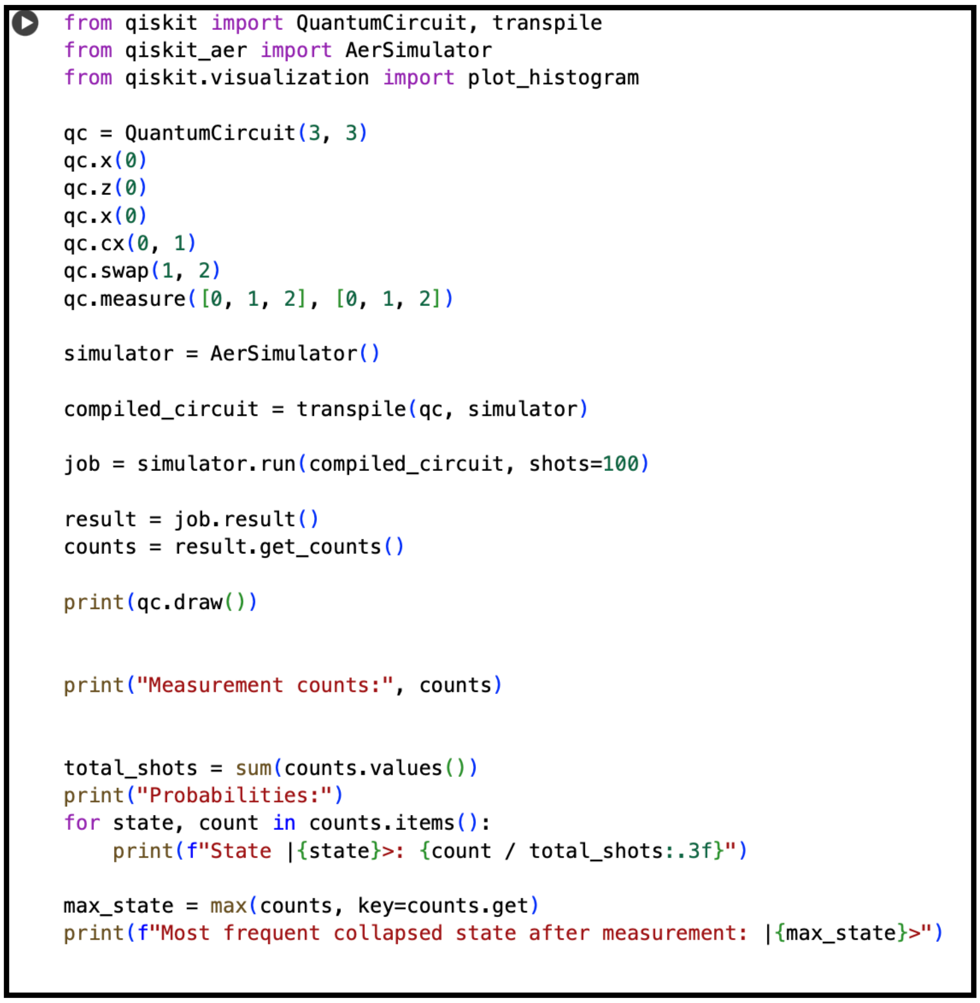
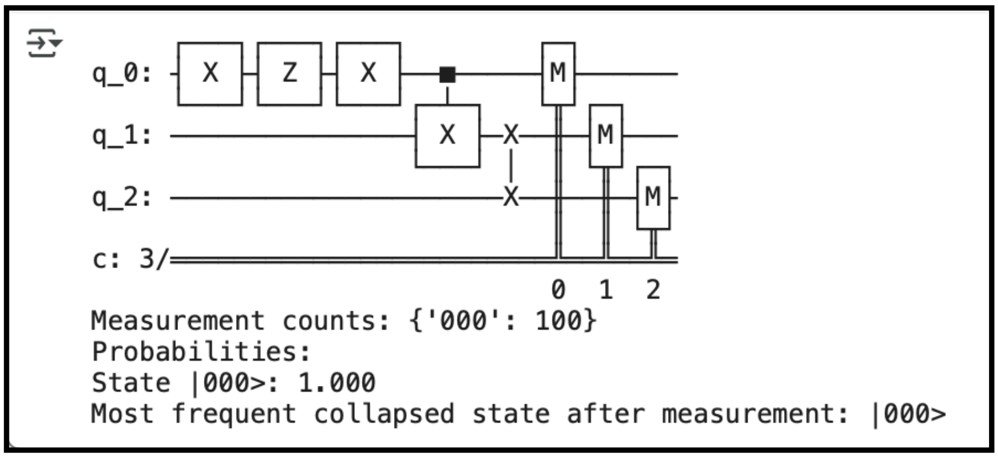
EXPERIMENT NO.5

Case-1:



```python
from qiskit import QuantumCircuit, transpile
from qiskit_aer import AerSimulator
from qiskit.visualization import plot_histogram

qc = QuantumCircuit(3, 3)
qc.x(0)
qc.z(0)
qc.x(0)
qc.cx(0, 1)
qc.swap(1, 2)
qc.measure([0, 1, 2], [0, 1, 2])

simulator = AerSimulator()

compiled_circuit = transpile(qc, simulator)

job = simulator.run(compiled_circuit, shots=100)

result = job.result()
counts = result.get_counts()

print(qc.draw())


print("Measurement counts:", counts)


total_shots = sum(counts.values())
print("Probabilities:")
for state, count in counts.items():
    print(f"State |{state}>: {count / total_shots:.3f}")

max_state = max(counts, key=counts.get)
print(f"Most frequent collapsed state after measurement: |{max_state}>")
```

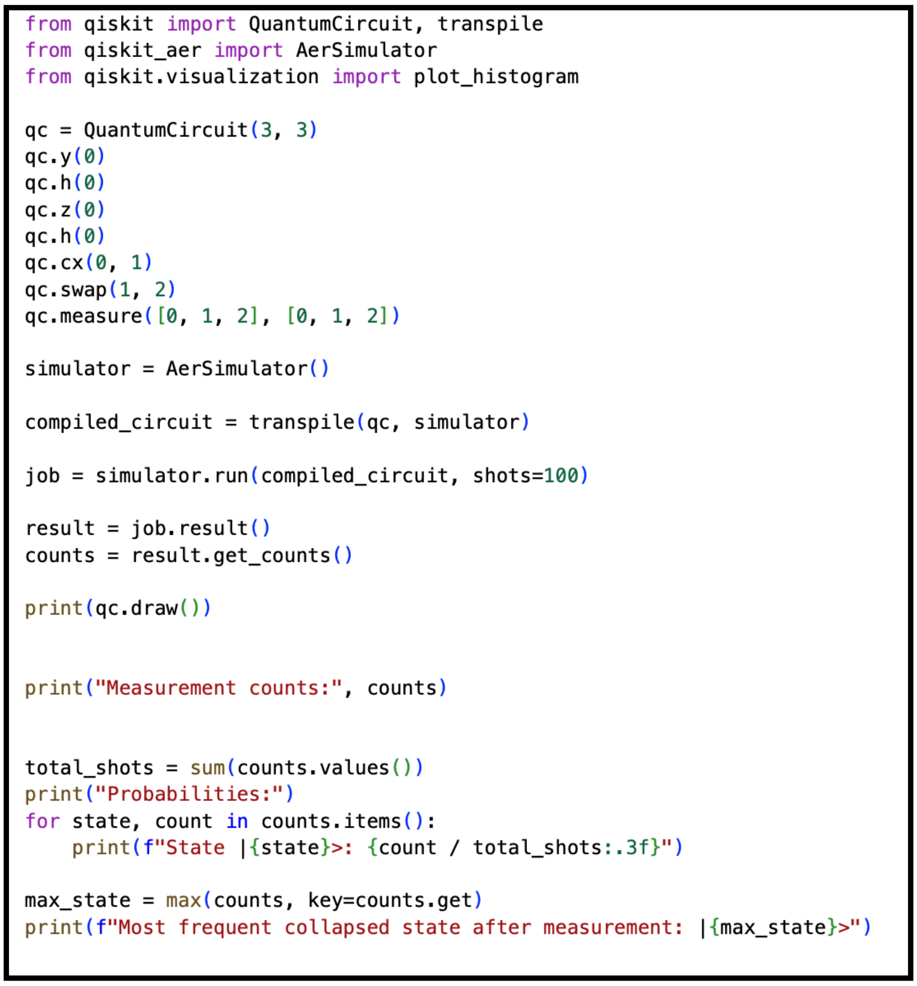
OUTPUT



```
     ┌───┐┌───┐┌───┐        ┌─┐      
q_0: ┤ X ├┤ Z ├┤ X ├──■─────┤M├──────
     └───┘└───┘└───┘┌─┴─┐   └╥┘┌─┐   
q_1: ───────────────┤ X ├─X──╫─┤M├───
                    └───┘ │  ║ └╥┘┌─┐
q_2: ─────────────────────X──╫──╫─┤M├
                             ║  ║ └╥┘
c: 3/════════════════════════╩══╩══╩═
                             0  1  2 
Measurement counts: {'000': 100}
Probabilities:
State |000>: 1.000
Most frequent collapsed state after measurement: |000>
```

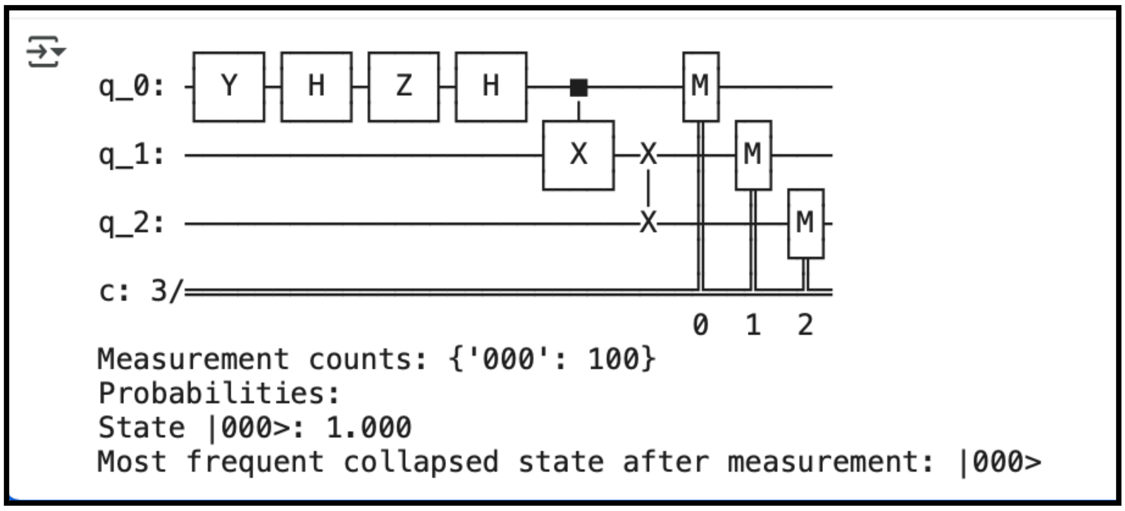
EXPERIMENT NO.5

Case-2:



```python
from qiskit import QuantumCircuit, transpile
from qiskit_aer import AerSimulator
from qiskit.visualization import plot_histogram

qc = QuantumCircuit(3, 3)
qc.y(0)
qc.h(0)
qc.z(0)
qc.h(0)
qc.cx(0, 1)
qc.swap(1, 2)
qc.measure([0, 1, 2], [0, 1, 2])

simulator = AerSimulator()

compiled_circuit = transpile(qc, simulator)

job = simulator.run(compiled_circuit, shots=100)

result = job.result()
counts = result.get_counts()

print(qc.draw())


print("Measurement counts:", counts)


total_shots = sum(counts.values())
print("Probabilities:")
for state, count in counts.items():
    print(f"State |{state}>: {count / total_shots:.3f}")

max_state = max(counts, key=counts.get)
print(f"Most frequent collapsed state after measurement: |{max_state}>")
```

OUTPUT

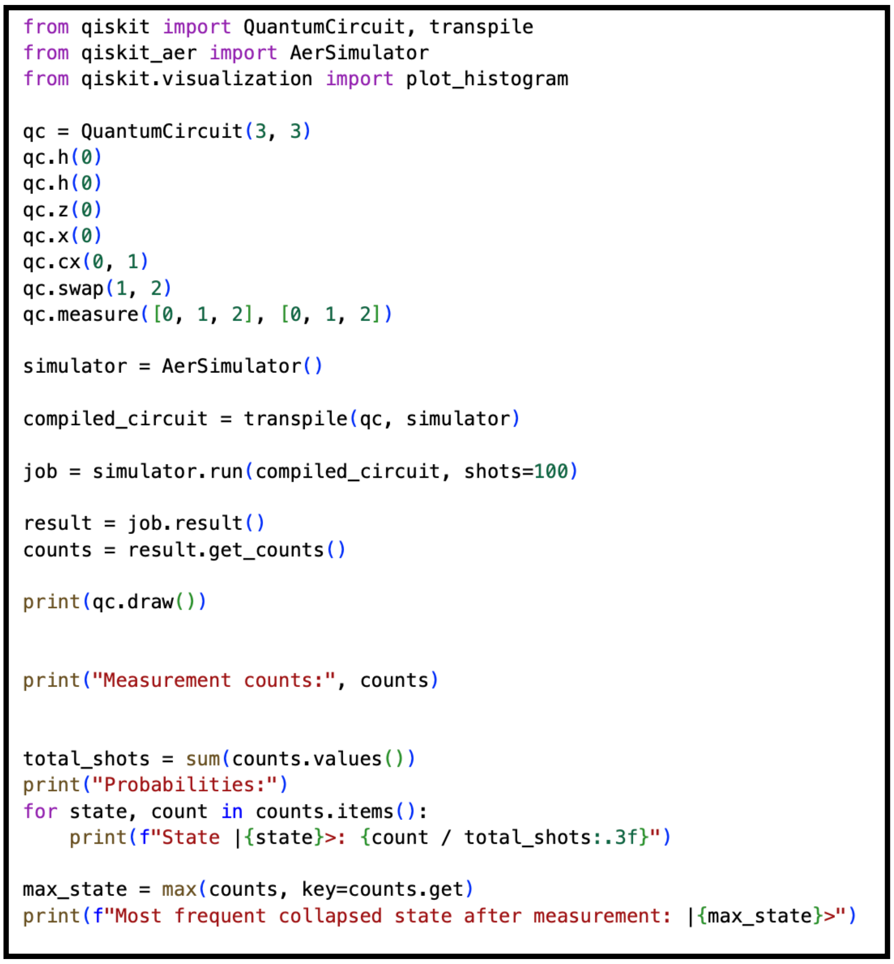


```
     ┌───┐┌───┐┌───┐┌───┐        ┌─┐      
q_0: ┤ Y ├┤ H ├┤ Z ├┤ H ├──■─────┤M├──────
     └───┘└───┘└───┘└───┘┌─┴─┐   └╥┘┌─┐   
q_1: ────────────────────┤ X ├─X──╫─┤M├───
                         └───┘ │  ║ └╥┘┌─┐
q_2: ──────────────────────────X──╫──╫─┤M├
                                  ║  ║ └╥┘
c: 3/═════════════════════════════╩══╩══╩═
                                  0  1  2 
Measurement counts: {'000': 100}
Probabilities:
State |000>: 1.000
Most frequent collapsed state after measurement: |000>
```

Case

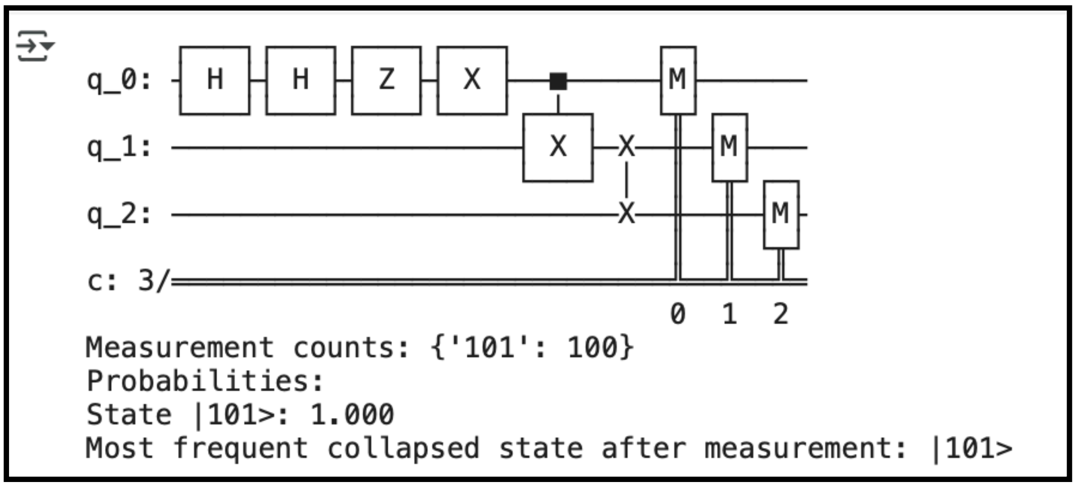
EXPERIMENT NO.5

CASE 3 :



```python
from qiskit import QuantumCircuit, transpile
from qiskit_aer import AerSimulator
from qiskit.visualization import plot_histogram

qc = QuantumCircuit(3, 3)
qc.h(0)
qc.h(0)
qc.z(0)
qc.x(0)
qc.cx(0, 1)
qc.swap(1, 2)
qc.measure([0, 1, 2], [0, 1, 2])

simulator = AerSimulator()

compiled_circuit = transpile(qc, simulator)

job = simulator.run(compiled_circuit, shots=100)

result = job.result()
counts = result.get_counts()

print(qc.draw())


print("Measurement counts:", counts)


total_shots = sum(counts.values())
print("Probabilities:")
for state, count in counts.items():
    print(f"State |{state}>: {count / total_shots:.3f}")

max_state = max(counts, key=counts.get)
print(f"Most frequent collapsed state after measurement: |{max_state}>")
```

OUTPUT:



```
     ┌───┐┌───┐┌───┐┌───┐     ┌─┐      
q_0: ┤ H ├┤ H ├┤ Z ├┤ X ├──■──┤M├──────
     └───┘└───┘└───┘└───┘┌─┴─┐└╥┘┌─┐   
q_1: ────────────────────┤ X ├─X──╫─┤M├───
                         └───┘ │  ║ └╥┘┌─┐
q_2: ──────────────────────────X──╫──╫─┤M├
                                  ║  ║ └╥┘
c: 3/═════════════════════════════╩══╩══╩═
                                  0  1  2 
Measurement counts: {'101': 100}
Probabilities:
State |101>: 1.000
Most frequent collapsed state after measurement: |101>
```