Date:

TASK 5: CNOT Gate and Quantum Teleportation

Aim: To simulate a CNOT gate and implement a simplified quantum teleportation protocol using Qiskit.

1 Mathematical Model of the CNOT Gate

- The **CNOT (Controlled-NOT)** gate is a two-qubit quantum gate that flips the target qubit if and only if the control qubit is in state $|1\rangle$.
- Computational basis ordering: $|00\rangle, |01\rangle, |10\rangle, |11\rangle$ with first qubit = control(q0), second qubit = target(q1).

1.1 Matrix Representation

The CNOT gate is represented by the following unitary matrix

$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

1.2 CNOT Gate Truth Table

Control (q0)	Target (q1)	Output (q0, q1)	Explanation
0	0	0, 0	Control is 0 → target unchanged
0	1	0, 1	Control is 0 → target unchanged
1	0	1, 1	Control is 1 → target flips (0→1)
1	1	1, 0	Control is 1 → target flips (1→0)

1.3 Effect on Basis States

- If **control = $|0\rangle$**, target remains unchanged.
- If **control = $|1\rangle$**, target flips (X-gate applied).

2 Algorithm for CNOT Gate Implementation

1. **Initialize** a quantum circuit with **2 qubits** and **2 classical bits**.
2. **Prepare input states** (e.g., test all possible combinations: $|00\rangle$, $|01\rangle$, $|10\rangle$, $|11\rangle$).
3. **Apply CNOT gate** (control qubit = q0, target qubit = q1).
4. **Measure** the qubits and store results in classical bits.
5. **Simulate** the circuit using Qiskit's Aer simulator.
6. **Plot** the measurement outcomes.

3 Program

```
from qiskit import QuantumCircuit
from qiskit_aer import Aer
from qiskit.visualization import plot_histogram
import matplotlib.pyplot as plt

def cnot_circuit(input_state):
    """
     Creates and simulates a CNOT circuit for a given input state.
    Args:
        input_state (str): '00', '01', '10', or '11'
    """
    qc = QuantumCircuit(2, 2)  # 2 qubits, 2 classical bits

    # Prepare input state
    if input_state[0] == '1':
        qc.x(0)  # Set q0 to |1⟩
    if input_state[1] == '1':
        qc.x(1)  # Set q1 to |1⟩

    # Apply CNOT (q0=control, q1=target)
    qc.cx(0, 1)

    # Measure qubits
    qc.measure([0, 1], [0, 1])

    # Simulate
    simulator = Aer.get_backend('qasm_simulator')
    result = simulator.run(qc, shots=1000).result()
    counts = result.get_counts(qc)

    # Plot results
    print(f"\nCNOT Gate Test | Input: |{input_state}⟩")
    print("Circuit Diagram:")
```

```
    print(qc.draw(output='text'))
    plot_histogram(counts)
    plt.show()

# Test all possible inputs
for state in ['00', '01', '10', '11']:
    cnot_circuit(state)
```

4. Mathematical Model for Quantum Teleportation

Quantum teleportation enables transferring an unknown quantum state from Alice to Bob using:

1. **Entanglement** (shared Bell pair)
2. **Classical communication** (2 bits)
3. **Quantum operations** (CNOT, Hadamard, measurements)

4.1 Initial Setup

- Alice has qubit

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \quad (|\alpha|^2 + |\beta|^2 = 1)$$

- Alice and Bob share an entangled Bell pair

$$|\Phi^+\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}$$

4.2 Step-by-Step State Evolution

1. **Combined state**

$$|\psi\rangle \otimes |\Phi^+\rangle = \frac{\alpha|0\rangle(|00\rangle + |11\rangle) + \beta|1\rangle(|00\rangle + |11\rangle)}{\sqrt{2}}$$

2. **Alice applies CNOT (q0 → q1)**

$$\frac{\alpha|0\rangle(|00\rangle + |11\rangle) + \beta|1\rangle(|10\rangle + |01\rangle)}{\sqrt{2}}$$

3. **Alice applies Hadamard to q0**

$$\frac{1}{2}\left[|00\rangle(\alpha|0\rangle+\beta|1\rangle)+|01\rangle(\alpha|1\rangle+\beta|0\rangle)+|10\rangle(\alpha|0\rangle-\beta|1\rangle)+|11\rangle(\alpha|1\rangle-\beta|0\rangle)\right]$$

4. **Alice measures q0 & q1** → gets one of 4 classical outcomes (00, 01, 10, 11)
5. **Bob applies corrections**
   - **00**: Do nothing
   - **01**: Apply X gate
   - **10**: Apply Z gate
   - **11**: Apply X then Z

Final state at Bob's qubit

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

5 Algorithm for Quantum Teleportation Implementation

1. **Initialize** 3-qubit circuit (Alice's q0, shared q1, Bob's q2) + 2 classical bits
2. **Prepare** Alice's qubit (e.g., $|1\rangle$ via X gate)
3. **Create Bell pair** between q1 & q2 (H + CNOT)
4. **Teleportation protocol**
   - CNOT(q0, q1)
   - H(q0)
   - Measure q0 & q1 → store in classical bits

5. **Bob's corrections**
   - Apply X if c1=1
   - Apply Z if c0=1

6. **Verify** by measuring Bob's qubit

6 Program for Quantum Teleportation Implementation

```
from qiskit import QuantumCircuit
from qiskit_aer import Aer
from qiskit.visualization import plot_histogram
import matplotlib.pyplot as plt

# Create circuit
qc = QuantumCircuit(3, 2)  # 3 qubits, 2 classical bits
```

```
# Step 1: Prepare Alice's state (|1⟩ for demo)
qc.x(0)  # Comment out to teleport |0⟩
qc.barrier()

# Step 2: Create Bell pair (q1 & q2)
qc.h(1)
qc.cx(1, 2)
qc.barrier()

# Step 3: Teleportation protocol
qc.cx(0, 1)
qc.h(0)
qc.barrier()

# Step 4: Measure Alice's qubits
qc.measure([0,1], [0,1])
qc.barrier()

# Step 5: Bob's corrections
qc.cx(1, 2)  # X if c1=1
qc.cz(0, 2)  # Z if c0=1

# Step 6: Measure Bob's qubit
qc.measure(2, 0)  # Overwrite c0 for verification

# Draw circuit
print("Teleportation Circuit:")
print(qc.draw(output='text'))

# Simulate
simulator = Aer.get_backend('qasm_simulator')
result = simulator.run(qc, shots=1000).result()
counts = result.get_counts(qc)

# Results
print("\nMeasurement results:")
print(counts)
plot_histogram(counts)
plt.show()
```

7. Result

This work illustrates the implementation, simulation, and verification of the CNOT gate using Qiskit, followed by the construction of a complete quantum teleportation protocol. The protocol is validated through simulation, confirming the accurate transfer of an arbitrary quantum state using entanglement and classical communication.