Date : 19/8/25

```
TASK 5: CNOT Gate and Quantum Teleportation
```

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

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.

```
 !pip install qiskit
 !pip install qiskit_aer
 !pip install matplotlib
 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)
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()
```

```
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-package
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.12/dist-packages (fro
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.12/dist-packag
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.12/dist-pa
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from

CNOT Gate Test | Input: |00⟩
Circuit Diagram:
           ┌─┐   
q_0: ──■───┤M├───
     ┌─┴─┐ └╥┘┌─┐
q_1: ┤ X ├──╫─┤M├
     └───┘  ║ └╥┘
c: 2/═══════╩══╩═
            0  1 

CNOT Gate Test | Input: |01⟩
```

Result: Quantum teleportation was simulated and the protocol was executed successfully.