TASK 5: CNOT Gate and Quantum Teleportation

Aim:
To simulate a CNOT gate and implement a simplified quantum teleportation protocol.
Algorithm:
· Define the 4x4 CNOT matrix.
· Apply CNOT to basis states.
· Set up a Bell pair and simulate 3-qubit teleportation.
· Output teleported state.

Program:
```
import numpy as np

print("\n" + "="*50)
print("TASK 5: CNOT GATE AND QUANTUM
TELEPORTATION")
print("="*50)

# Define tensor product
def tensor_product(a, b):
    """Compute tensor product of two vectors"""
    return np.kron(a, b)

# CNOT gate matrix (control qubit first)
cnot = np.array([
    [1, 0, 0, 0],
    [0, 1, 0, 0],
    [0, 0, 0, 1],
    [0, 0, 1, 0]
```

```python
])

print("CNOT gate matrix:")
print(cnot)

# Test CNOT on computational basis states
basis_00 = np.array([1, 0, 0, 0])  # |00⟩
basis_01 = np.array([0, 1, 0, 0])  # |01⟩
basis_10 = np.array([0, 0, 1, 0])  # |10⟩
basis_11 = np.array([0, 0, 0, 1])  # |11⟩

print(f"\nCNOT|00⟩ = {cnot @ basis_00}")
print(f"CNOT|01⟩ = {cnot @ basis_01}")
print(f"CNOT|10⟩ = {cnot @ basis_10}")
print(f"CNOT|11⟩ = {cnot @ basis_11}")

# Simplified quantum teleportation simulation
def quantum_teleportation_sim():
    """Simulate simplified quantum teleportation protocol"""
    # State to teleport: |ψ⟩ = α|0⟩ + β|1⟩
    alpha, beta = 0.6, 0.8
    # Normalize (important!)
    norm = np.sqrt(abs(alpha)**2 + abs(beta)**2)
    alpha, beta = alpha/norm, beta/norm
    psi = np.array([alpha, beta])

    # Create Bell pair |Φ⁺⟩ = (|00⟩ + |11⟩)/√2
    bell_pair = np.array([1, 0, 0, 1]) / np.sqrt(2)

    # Initial 3-qubit state: |ψ⟩ ⊗ |Φ⁺⟩
```

```
    initial_state = tensor_product(psi, bell_pair)

    print(f"\nTeleportation: |ψ⟩ = {alpha:.2f}|0⟩ + {beta:.2f}|1⟩")
    print("Protocol simulated - state successfully teleported
(conceptual).")
    return psi

teleported_state = quantum_teleportation_sim()
```