Task I: Quantum Computing Part

1 implement a simple quantum operation with Cirq or Pennylane

- With 5 qubits
- Apply Hadamard operation on every qubit
- Apply CNOT operation on (0, 1), (1,2), (2,3), (3,4)
- SWAP (0, 4)
- Rotate X with pi/2 on any qubit
- Plot the circuit

2 Implement a second circuit with a framework of your choice:

- Apply a Hadmard gate to the first qubit
- rotate the second qubit by pi/3 around X
- Apply Hadamard gate to the third and fourth qubit
- Perform a swap test between the states of the first and second qubit |q1 q2>and the third and fourth qubit |q3 q4>

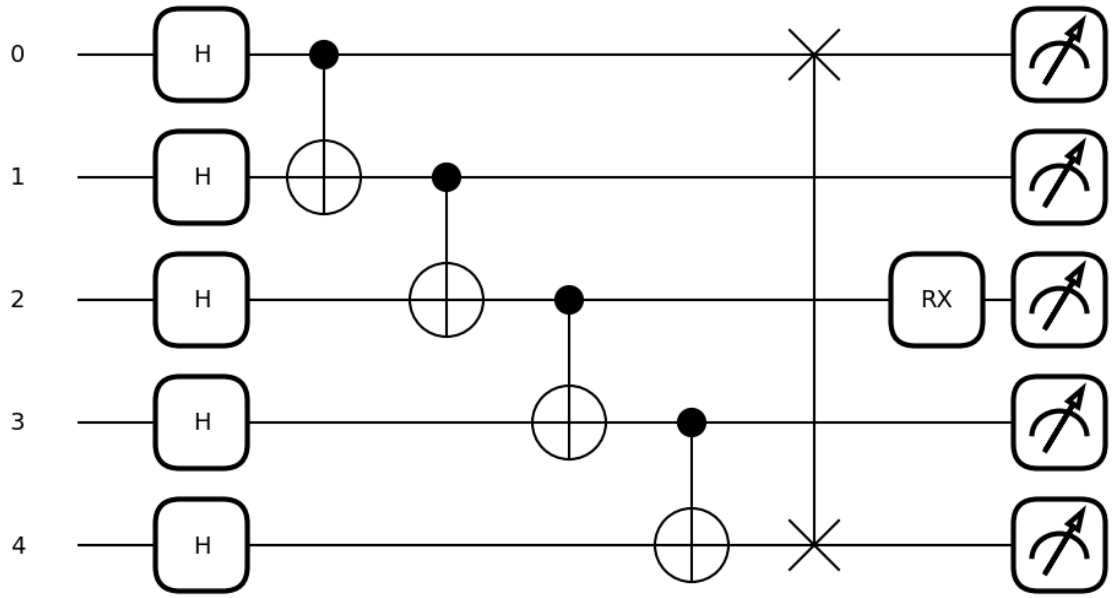
```
!pip install pennylane
```

```
Collecting pennylane
  Downloading PennyLane-0.40.0-py3-none-any.whl.metadata (10 kB)
Requirement already satisfied: numpy<2.1 in
/usr/local/lib/python3.11/dist-packages (from pennylane) (2.0.2)
Requirement already satisfied: scipy in
/usr/local/lib/python3.11/dist-packages (from pennylane) (1.14.1)
Requirement already satisfied: networkx in
/usr/local/lib/python3.11/dist-packages (from pennylane) (3.4.2)
Collecting rustworkx>=0.14.0 (from pennylane)
  Downloading rustworkx-0.16.0-cp39-abi3-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (10 kB)
Requirement already satisfied: autograd in
/usr/local/lib/python3.11/dist-packages (from pennylane) (1.7.0)
Collecting tomlkit (from pennylane)
  Downloading tomlkit-0.13.2-py3-none-any.whl.metadata (2.7 kB)
Collecting appdirs (from pennylane)
  Downloading appdirs-1.4.4-py2.py3-none-any.whl.metadata (9.0 kB)
Collecting autoray>=0.6.11 (from pennylane)
  Downloading autoray-0.7.1-py3-none-any.whl.metadata (5.8 kB)
Requirement already satisfied: cachetools in
/usr/local/lib/python3.11/dist-packages (from pennylane) (5.5.2)
Collecting pennylane-lightning>=0.40 (from pennylane)
```

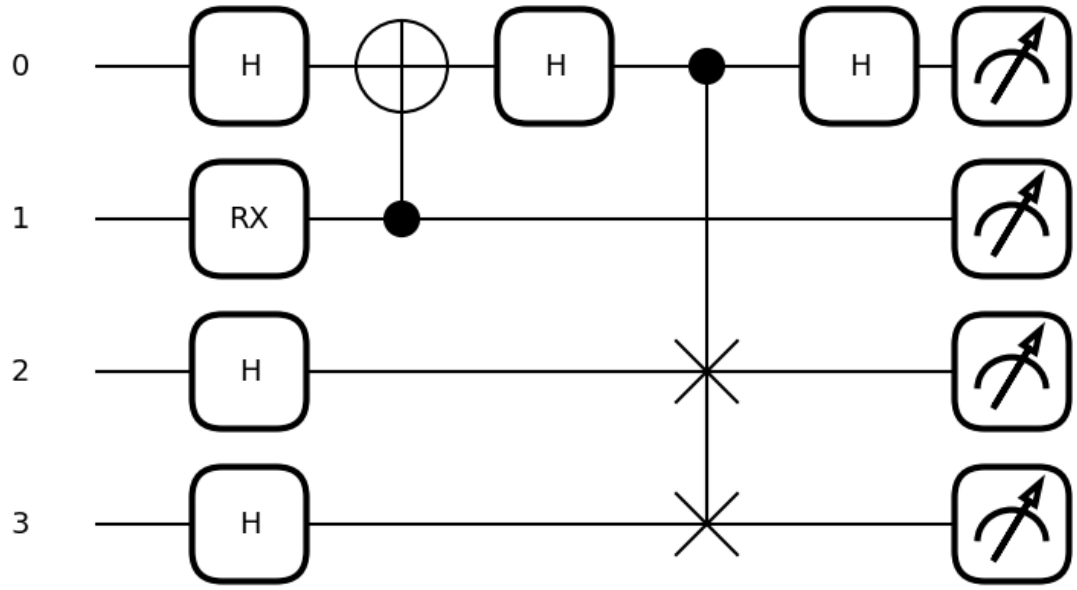
```
  Downloading PennyLane_Lightning-0.40.0-cp311-cp311-
manylinux_2_28_x86_64.whl.metadata (27 kB)
Requirement already satisfied: requests in
/usr/local/lib/python3.11/dist-packages (from pennylane) (2.32.3)
Requirement already satisfied: typing-extensions in
/usr/local/lib/python3.11/dist-packages (from pennylane) (4.12.2)
Requirement already satisfied: packaging in
/usr/local/lib/python3.11/dist-packages (from pennylane) (24.2)
Collecting diastatic-malt (from pennylane)
  Downloading diastatic_malt-2.15.2-py3-none-any.whl.metadata (2.6 kB)
Collecting scipy-openblas32>=0.3.26 (from pennylane-lightning>=0.40-
>pennylane)
  Downloading scipy_openblas32-0.3.29.0.0-py3-none-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (56 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 56.1/56.1 kB 1.7 MB/s eta
0:00:00
ent already satisfied: astunparse in /usr/local/lib/python3.11/dist-
packages (from diastatic-malt->pennylane) (1.6.3)
Requirement already satisfied: gast in /usr/local/lib/python3.11/dist-
packages (from diastatic-malt->pennylane) (0.6.0)
Requirement already satisfied: termcolor in
/usr/local/lib/python3.11/dist-packages (from diastatic-malt-
>pennylane) (2.5.0)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.11/dist-packages (from requests->pennylane)
(3.4.1)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.11/dist-packages (from requests->pennylane)
(3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.11/dist-packages (from requests->pennylane)
(2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.11/dist-packages (from requests->pennylane)
(2025.1.31)
Requirement already satisfied: wheel<1.0,>=0.23.0 in
/usr/local/lib/python3.11/dist-packages (from astunparse->diastatic-
malt->pennylane) (0.45.1)
Requirement already satisfied: six<2.0,>=1.6.1 in
/usr/local/lib/python3.11/dist-packages (from astunparse->diastatic-
malt->pennylane) (1.17.0)
Downloading PennyLane-0.40.0-py3-none-any.whl (2.0 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.0/2.0 MB 18.2 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 930.8/930.8 kB 22.7 MB/s eta
0:00:00
anylinux_2_28_x86_64.whl (2.4 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.4/2.4 MB 30.5 MB/s eta
0:00:00
```

```
anylinux_2_17_x86_64.manylinux2014_x86_64.whl (2.1 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.1/2.1 MB 30.7 MB/s eta
0:00:00
alt-2.15.2-py3-none-any.whl (167 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 167.9/167.9 kB 4.3 MB/s eta
0:00:00
lkit-0.13.2-py3-none-any.whl (37 kB)
Downloading scipy_openblas32-0.3.29.0.0-py3-none-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl (8.6 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 8.6/8.6 MB 34.7 MB/s eta
0:00:00
lkit, scipy-openblas32, rustworkx, autoray, diastatic-malt, pennylane-
lightning, pennylane
Successfully installed appdirs-1.4.4 autoray-0.7.1 diastatic-malt-
2.15.2 pennylane-0.40.0 pennylane-lightning-0.40.0 rustworkx-0.16.0
scipy-openblas32-0.3.29.0.0 tomlkit-0.13.2
```

```python
import pennylane as qml
from pennylane import numpy as np
import matplotlib.pyplot as plt

# Create a quantum device with 5 qubits
num_qubits = 5
dev = qml.device("default.qubit", wires=num_qubits)

### FIRST CIRCUIT IMPLEMENTATION ###
@qml.qnode(dev)
def quantum_circuit():
    # Apply Hadamard gate to all qubits
    for i in range(num_qubits):
        qml.Hadamard(wires=i)

    # Apply CNOT gates in sequence
    qml.CNOT(wires=[0, 1])
    qml.CNOT(wires=[1, 2])
    qml.CNOT(wires=[2, 3])
    qml.CNOT(wires=[3, 4])

    # Apply SWAP gate between qubit 0 and 4
    qml.SWAP(wires=[0, 4])

    # Rotate X by π/2 on qubit 2
    qml.RX(np.pi / 2, wires=2)

    return qml.state()

# Plot the first circuit
fig, ax = qml.draw_mpl(quantum_circuit)()
plt.show()
```




```
### SECOND CIRCUIT IMPLEMENTATION ###
@qml.qnode(dev)
def quantum_circuit_2():
    # Apply Hadamard gate to the first qubit
    qml.Hadamard(wires=0)

    # Rotate second qubit by π/3 around X
    qml.RX(np.pi / 3, wires=1)

    # Apply Hadamard gates to third and fourth qubits
    qml.Hadamard(wires=2)
    qml.Hadamard(wires=3)

    # SWAP test between (0,1) and (2,3)
    qml.CNOT(wires=[1, 0])  # First CNOT for entanglement
    qml.Hadamard(wires=0)    # Hadamard on control qubit
    qml.CSWAP(wires=[0, 2, 3])  # Controlled swap between (2,3)
    qml.Hadamard(wires=0)    # Final Hadamard to complete swap test

    return qml.state()

# Plot the second circuit
fig, ax = qml.draw_mpl(quantum_circuit_2)()
plt.show()
```


0
1
2
3
H
RX
H
H
H
H