Maheen Shakeel

Hackathon Tasks Solutions

Problem 1 – Quantum Communication Simulator

Goal: Build a small simulation that demonstrates how quantum superposition and entanglement can be used to send and verify information securely between two users (Alice and Bob).

Tasks:

1. Create a pair of entangled qubits shared between Alice and Bob.
2. Allow Alice to apply operations (X, Z, H) on her qubit.
3. Bob measures his qubit – show how his result depends on Alice's operation.
4. Visualize results using Bloch spheres or histograms.

EXPLANATION for code implementation:

- we will begin by importing everything necessary for the code to work and visualize.
  CODE:

```
%matplotlib inline

from qiskit import QuantumCircuit, transpile
from qiskit.quantum_info import Statevector
from qiskit_aer import AerSimulator
from qiskit.visualization import plot_histogram , plot_bloch_multivector
import matplotlib.pyplot as plt
from qiskit.quantum_info import partial_trace
import numpy as np
```

  - the very first line is there to ensure Jupyter Notebook since I ran my code in VS and I wanted that and to display plots directly inside the notebook, right under the code cell that produced them.

```
%matplotlib inline
```

```
def alice(op):
    qop = QuantumCircuit(2, 2)
    qop.h(0)
    qop.cx(0, 1)
    qop.draw('mpl')
    if op == 'X':
        qop.x(0)
    elif op == 'Z':
        qop.z(0)
```

```
    elif op == 'H':
        qop.h(0)
    elif op is None:
        pass
    else:
        raise ValueError("Operation must be 'X', 'Z', 'H', or None")
    return qop
```

now here in this above snippet,

1. **Alice has started** with qubit |0⟩.
2. **Hadamard (H) gate** puts it into a **superposition** (50-50 of 0 and 1).
3. **CNOT gate** links Alice's qubit to Bob's (they become entangled.)
    - Now, measuring one determines the other.
4. Alice applies an operation (X, Z, H) to encode a message or alter the shared state.
    - This changes what Bob will see when he measures his qubit.

So, alice(op) builds the **quantum communication channel setup** , it's basically the base of the simulation showing how Alice's operation affects Bob's measurement.

```
def measure_and_visualize(op):
    circuit = alice(op)
    circuit.measure([0, 1], [0, 1])
    simulator = AerSimulator()
    job = simulator.run(circuit, shots=1000)
    result = job.result()
    counts = result.get_counts()
    print(f"Results when Alice applies {op or 'no operation'}:")
    print(counts)
    plot_histogram(counts)
    plt.show()

for operation in [None, 'X', 'Z', 'H']:
    measure_and_visualize(operation)
```

this part of the code is actually responsible for visualization oof all that we have done,

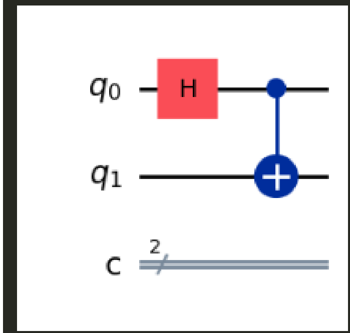
- The alice() function **creates the communication setup** (entanglement + operation).
- The measure_and_visualize() function runs the experiment and shows what happens when Bob measures.

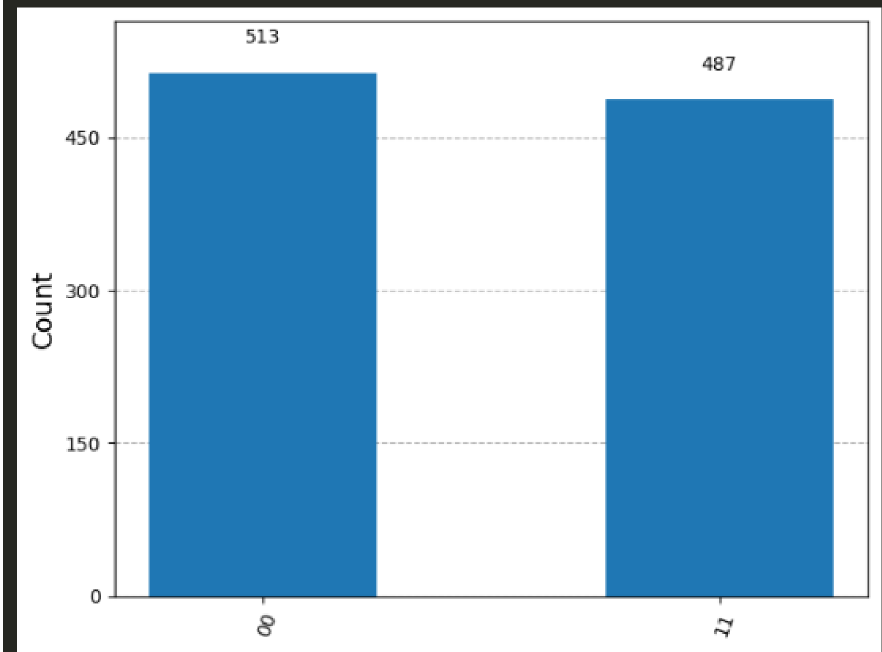
Maheen Shakeel

- The histograms wwe get are the visible proof of quantum entanglement and communication, basically they show how Alice's choice of operation changes Bob's results.

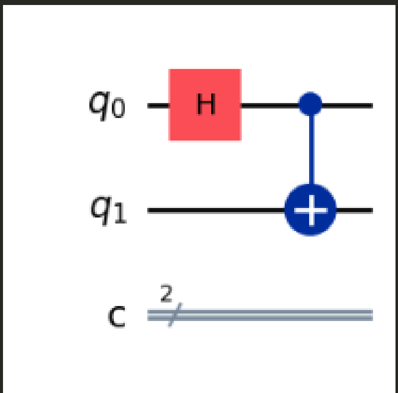
VISUAL SOLUTION OUTPUTS:

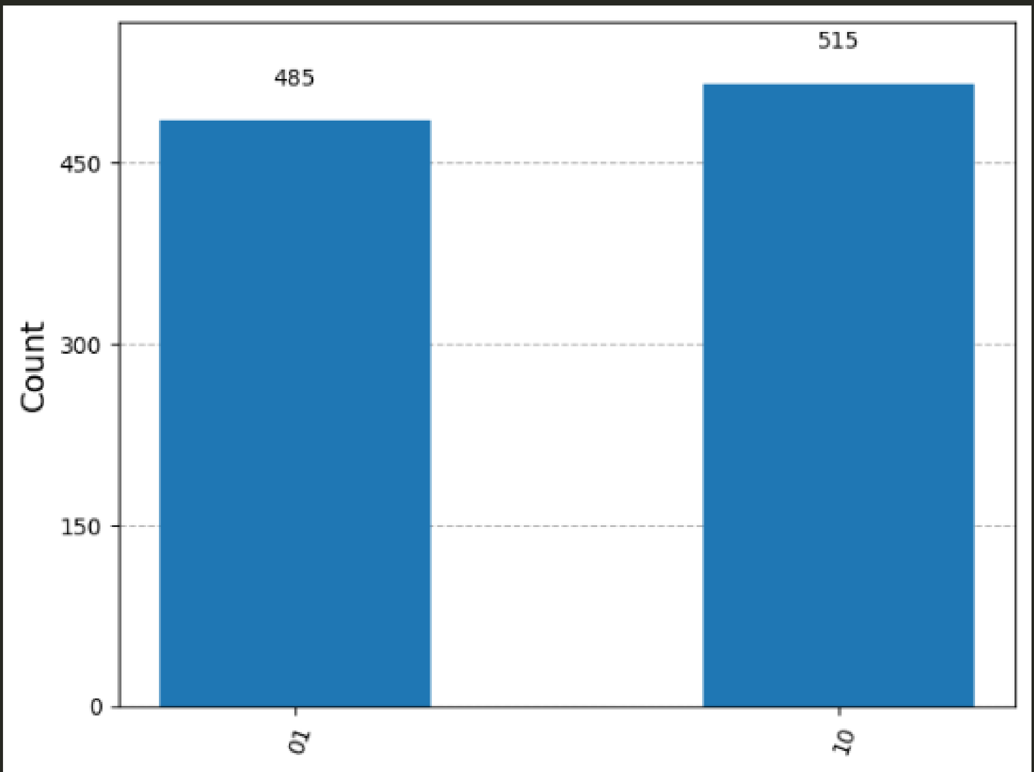
```
Results when Alice applies no operation:
{'11': 487, '00': 513}
```



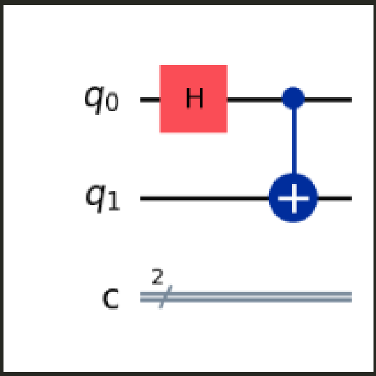


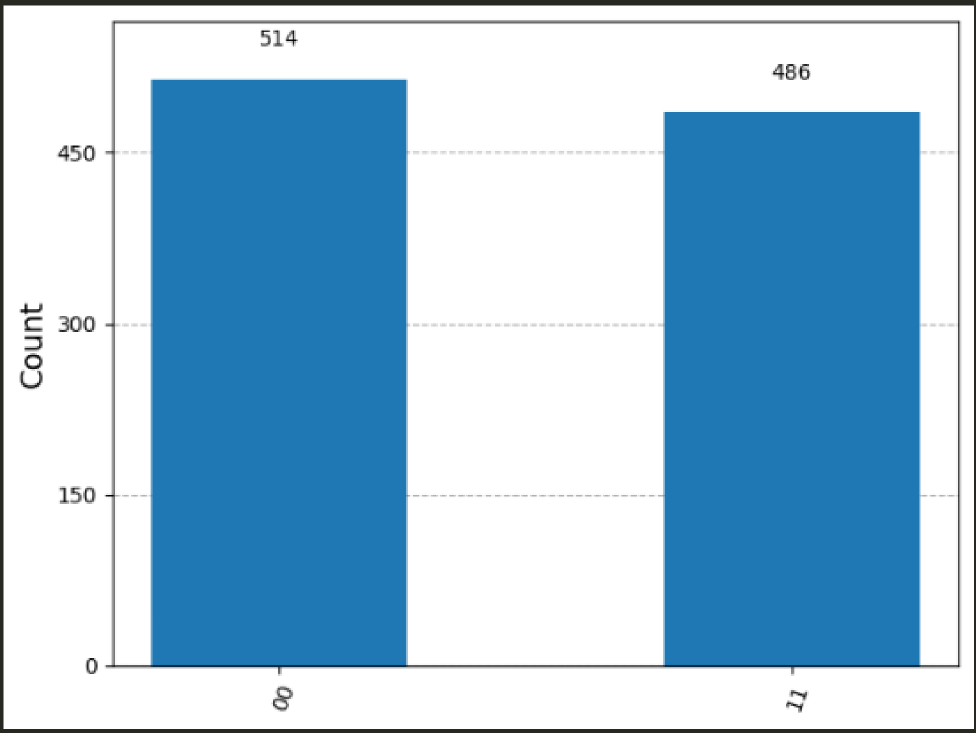
```
Results when Alice applies X:
{'10': 515, '01': 485}
```



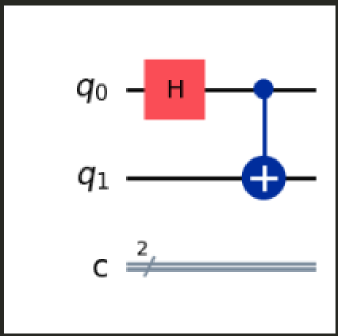


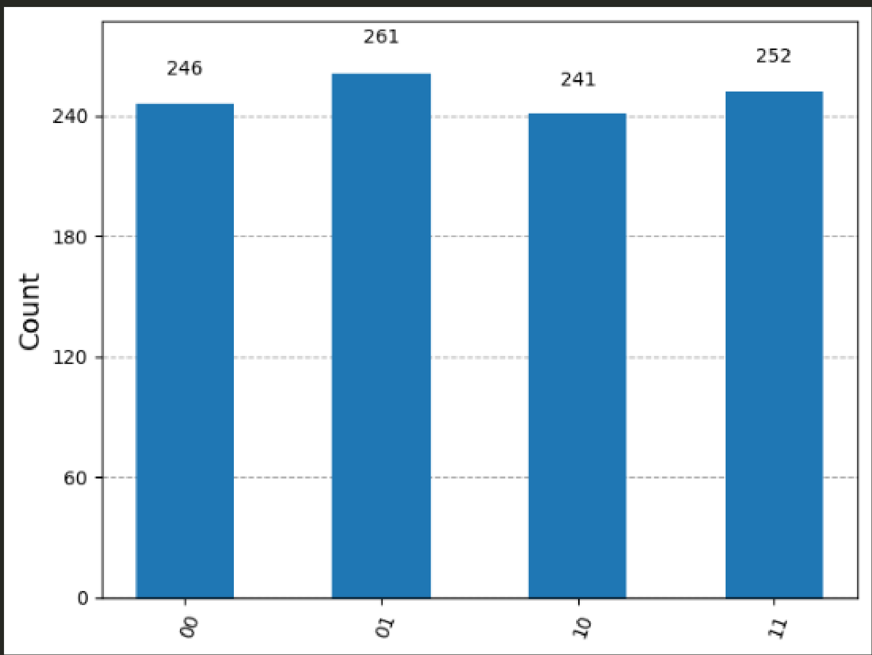
```
Results when Alice applies Z:
{'11': 486, '00': 514}
```





```
Results when Alice applies H:
{'10': 241, '11': 252, '00': 246, '01': 261}
```





Stretch Goal:

Extend to 3 qubits (Alice, Bob, Charlie) or simulate quantum teleportation.

EXPLANATION for code implementation:

```
def ex_alice(op):
    q = QuantumCircuit(3, 3)
    q.h(0)
    q.cx(0, 1)
    q.cx(0, 2)
```

ex_alice(op) creates a 3-qubit GHZ entangled system (meaning all three qubits will always measure the same, either all 1 or all 0).

```
if op == 'X':
        q.x(0)
    elif op == 'Z':
        q.z(0)
    elif op == 'H':
        q.h(0)
    elif op is None:
        pass
```

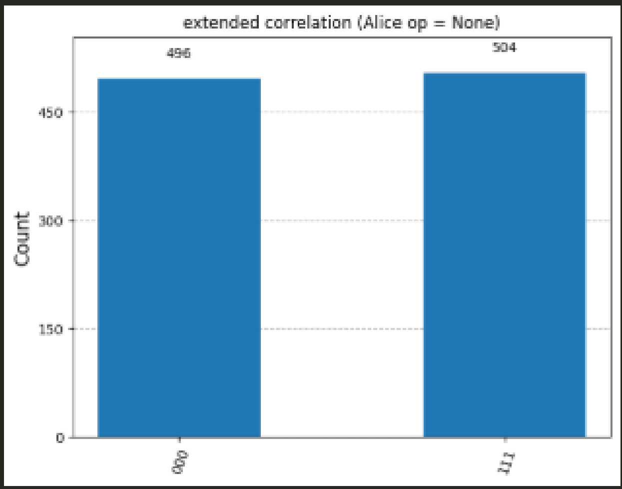
now here, Alice applies an operation on her own qubit (0):

- **X** → flips |0⟩↔|1⟩ → flips all correlations (000 ↔ 111 → 100 ↔ 011)
- **Z** → phase flip → doesn't change measurement counts but changes the overall phase
- **H** → puts Alice's qubit into a different basis → produces a more random spread in histogram
- **None** → keeps the perfect GHZ correlation.

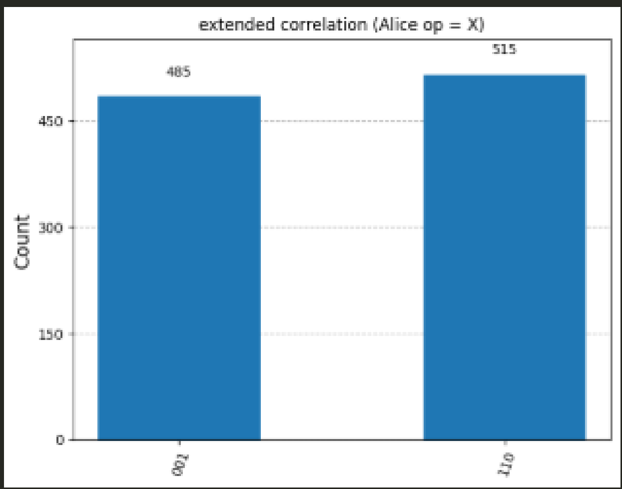
Whatever operation Alice applies changes the joint correlation pattern seen by all three when they measure.

VISUAL SOLUTION OUTPUTS:

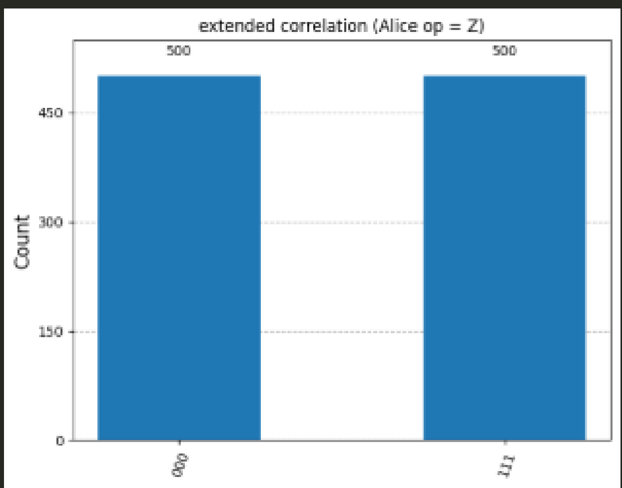
```
Results when Alice applies no operation:
{'111': 504, '000': 496}
```



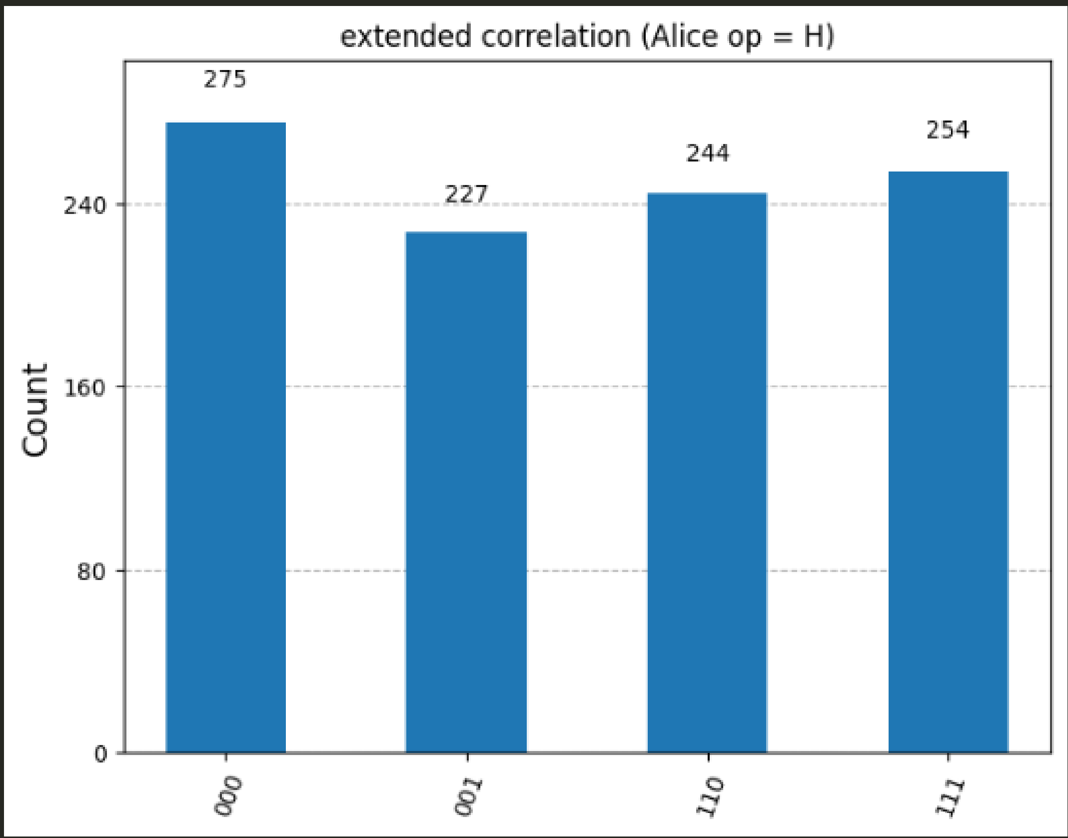

```
Results when Alice applies X:
{'001': 485, '110': 515}
```




```
Results when Alice applies Z:
{'000': 500, '111': 500}
```



```
Results when Alice applies H:
{'000': 275, '111': 254, '001': 227, '110': 244}
```



Problem 2 – Quantum Coin Game (Superposition)

Goal: Use quantum superposition to simulate a coin that is both heads and tails until measured.

Tasks:

1. Create a qubit initialized in |0■ (Heads).
2. Apply a Hadamard gate (H) to create a superposition (Heads and Tails).
3. Measure the qubit multiple times (e.g., 1000 shots) and show the distribution.
4. Add bias using RY(θ) instead of H to make the coin unfair.

EXPLANATION for code implementation:

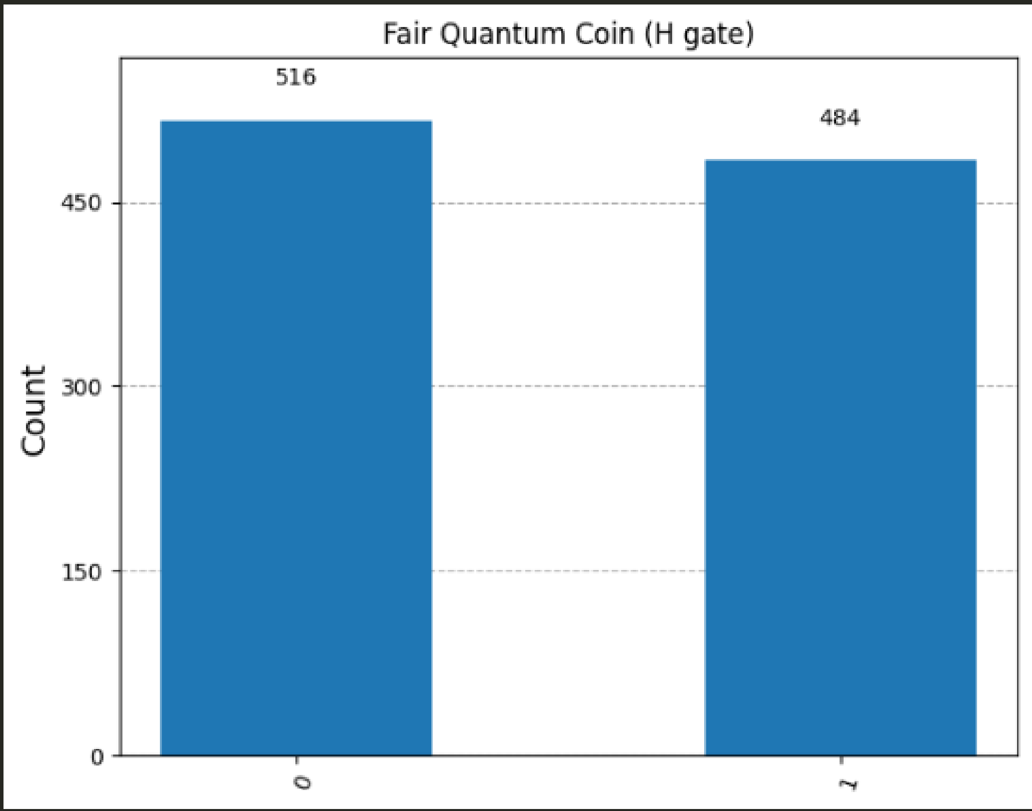
```
def quantum_coin(theta=None):
    q = QuantumCircuit(1, 1)
    if theta is None:
        q.h(0)
    else:
        q.ry(theta, 0)
    q.measure(0, 0)
    simulator = AerSimulator()
    job = simulator.run(q, shots=1000)
    result = job.result()
    counts = result.get_counts()
    return counts
```

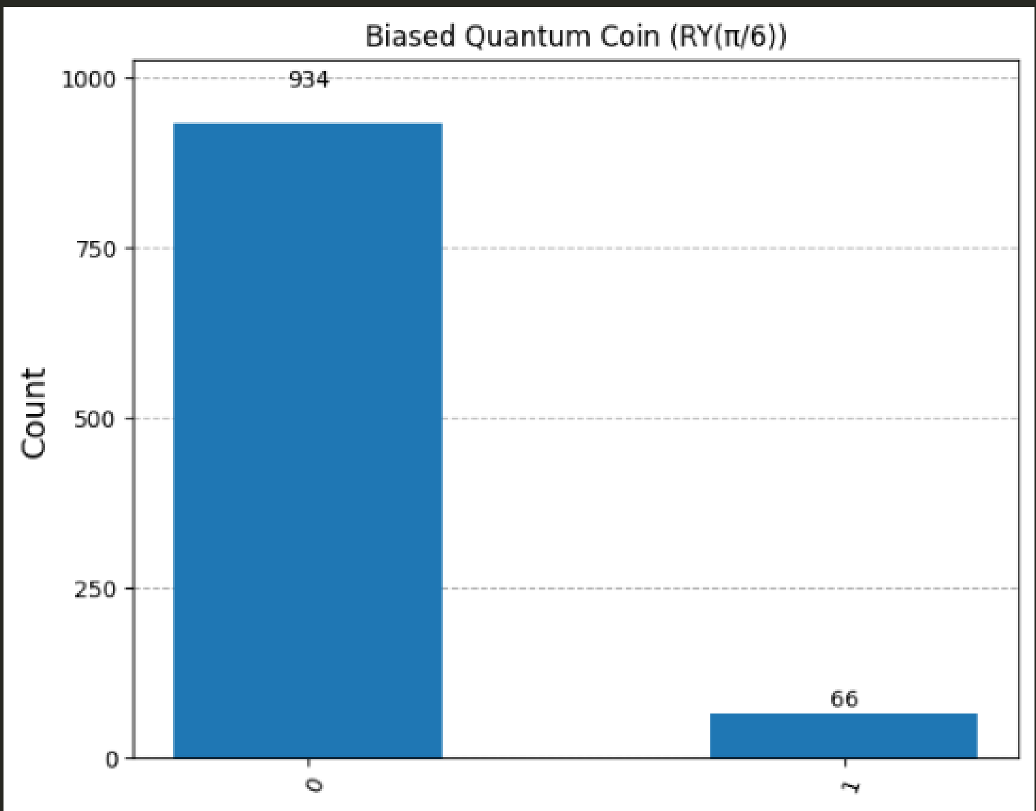
Basically, this function is explaining how the qubit starts in state |0⟩|0⟩|0⟩

- A Hadamard (H) or rotation (Ry) gate puts it into a superposition [representing a coin that is both heads and tails at once.]
- Then it's measured, collapsing randomly to 0 or 1 (heads or tails).
- The function runs many times (1000 "shots") to get the probability distribution of outcomes.

Then its visualized

VISUAL SOLUTION OUTPUTS:


Fair Quantum Coin (H gate)
516
484
Count
450
300
150
0
0
1


Biased Quantum Coin (RY(π/6))
934
66
Count
1000
750
500
250
0
0
1

Stretch Goal:

Overview of whats happening:

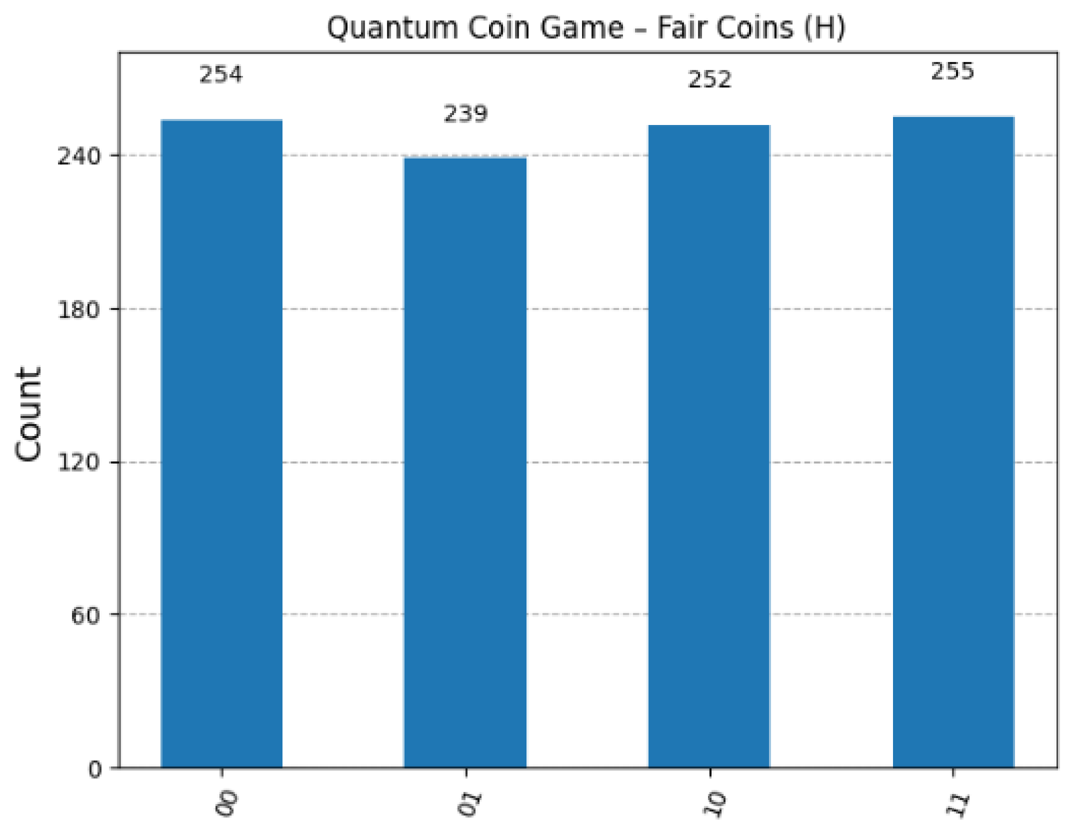
- If no theta is given → **Hadamard gate (H)** = fair coin (50/50 chance).
- If a specific θ is given → **rotation RY(θ)** = biased coin.

Each player can have a different bias:

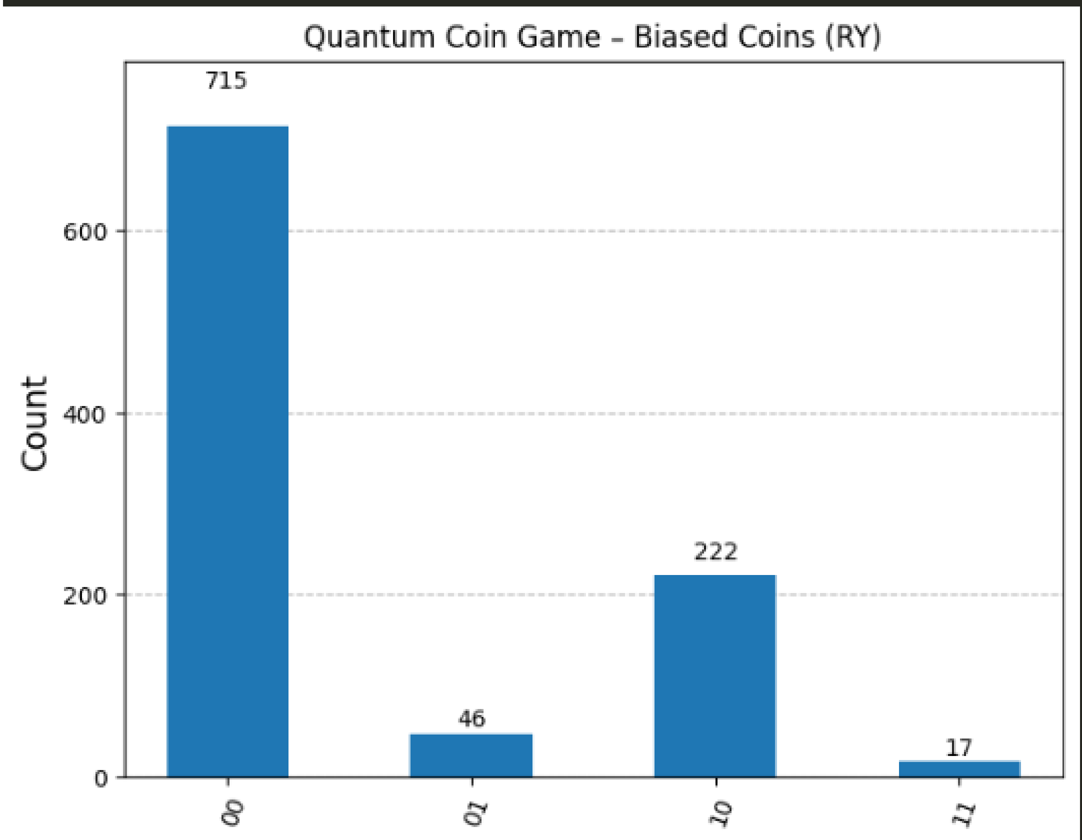
- Alice → θ_a
- Bob → θ_b

VISUAL SOLUTION OUTPUTS:

Maheen Shakeel



inner: Alice



Maheen Shakeel

Problem 3 – Quantum Correlation Explorer (Entanglement)

Goal: Visualize and analyze entanglement correlations between two qubits when measured in

different bases.

Tasks:

1. Create a 2-qubit Bell state (H + CNOT).
2. Measure both qubits and show 00 or 11 outcomes.
3. Apply different rotations (H, S, T) before measurement to change the measurement basis.
4. Measure again and show that correlation persists even with basis changes.

EXPLANATION for code implementation:

Creates a two-qubit entangled system (a Bell pair) and then explores how adding different single-qubit phase or Hadamard gates affects the measurement correlations between them.

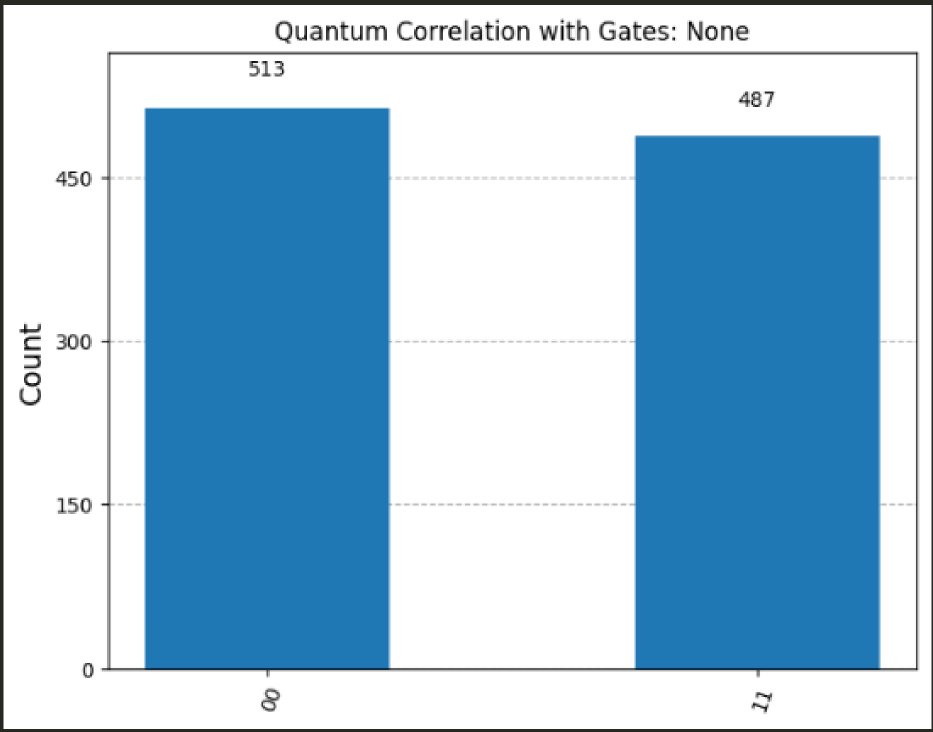
It's called a *correlation explorer* because we're literally observing how operations on both qubits change their joint measurement outcomes.

- If i measure qubit 0 as 0 → qubit 1 is guaranteed to be 0
- If i measure qubit 0 as 1 → qubit 1 is guaranteed to be 1

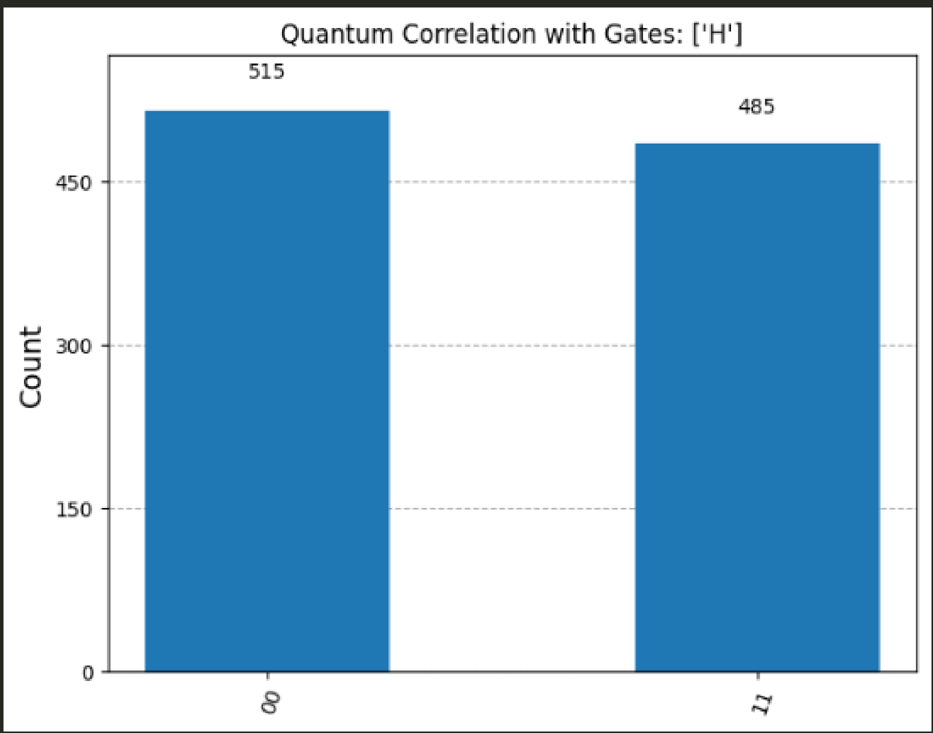
They are perfectly correlated even though the outcomes are random.

S and T are phase gates, and they are responsible for phase rotations that alter observable correlation.

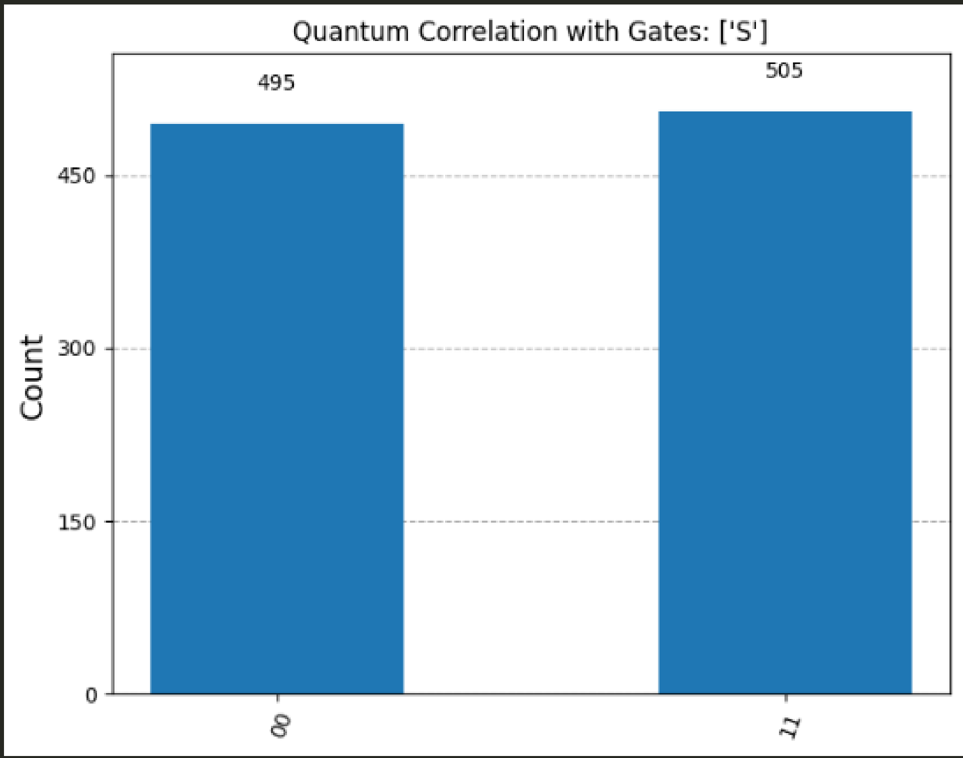
VISUAL SOLUTION OUTPUTS:



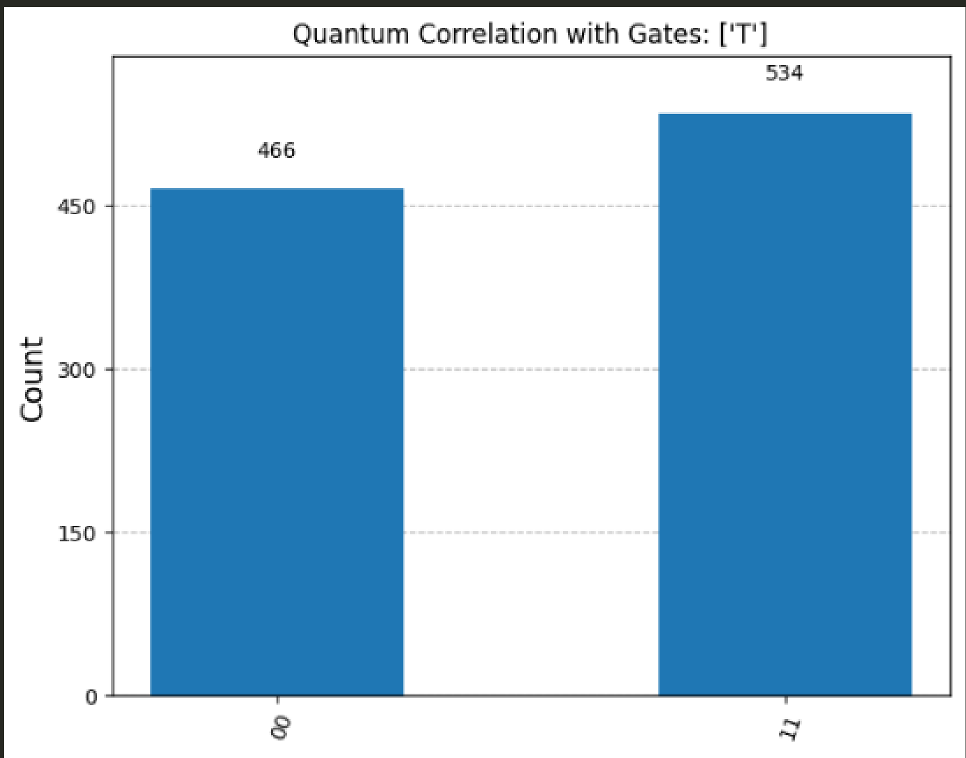

```
Gates applied: None
{'00': 513, '11': 487}
```



```
Gates applied: ['H']
{'11': 485, '00': 515}
```




```
Gates applied: ['S']
{'11': 505, '00': 495}
```



Stretch Goal:

Compare multiple Bell states (Φ■, Φ■, Ψ■) and calculate correlation coefficients.

EXPLANATION for code implementation:

1. Creates each of the four Bell states — $|\Phi^{+}\rangle$, $|\Phi^{-}\rangle$, $|\Psi^{+}\rangle$, and $|\Psi^{-}\rangle$.
2. Simulates measuring them.
3. Visualizes the measurement correlation for each state as a histogram.

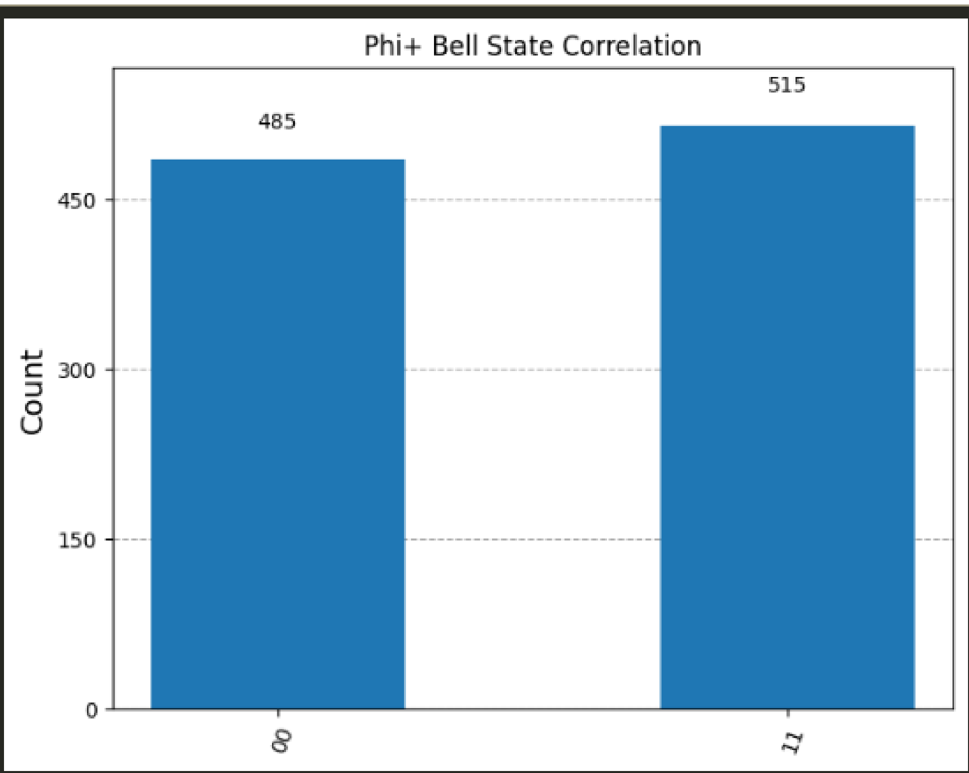
We're exploring how each Bell state behaves when we measure it in the computational (Z) basis.

Each gate combination transforms the initial $|\Phi^{+}\rangle$ into one of the other three Bell states.

When measured in the Z-basis:

- **Φ states** → perfectly correlated (both 0 or both 1)
- **Ψ states** → perfectly anti-correlated (one 0, one 1)

VISUAL SOLUTION OUTPUTS:




```
Phi+ state results:
{'11': 515, '00': 485}
```