Assignment 4

Problem Statement:

Tackle Noise with Error Correction

Objective:

1. Understand Tackle Noise with Error Correction

Outcome

Displays circuit for Tackle Noise with Error Correction.

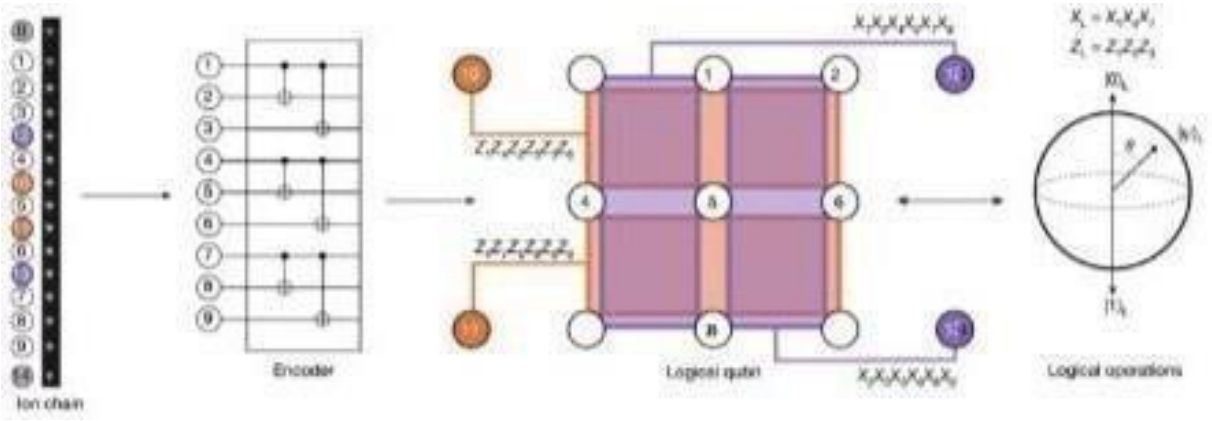
Software Requirement: Ubuntu OS, Quskit (Cloud based)

Theory:

Quantum error correction is theorised as essential to achieve fault tolerant quantum computing that can reduce the effects of noise on stored quantum information, faulty quantum gates, faulty quantum preparation, and faulty measurements. This would allow algorithms of greater circuit depth.
Noise is a major challenge in quantum computing, as it can cause errors in quantum computations. Quantum error correction (QEC) is a technique that can be used to tackle noise and improve the reliability of quantum computers.

QEC works by encoding a single logical quantum bit (qubit) into multiple physical qubits. This redundancy allows QEC to detect and correct errors that occur on the physical qubits.
There are a number of different QEC codes, each with its own strengths and weaknesses. Some QEC codes are more efficient, while others are more robust to noise.
QEC is still under development, but it has already been demonstrated in small-scale experiments. As QEC codes improve and become more efficient, they will be essential for building large-scale quantum computers.

QEC works by encoding a single logical quantum bit (qubit) into multiple physical qubits. This redundancy allows QEC to detect and correct errors that occur on the physical qubits.
There are a number of different QEC codes, each with its own strengths and weaknesses. Some QEC codes are more efficient, while others are more robust to noise.
QEC is still under development, but it has already been demonstrated in small-scale experiments. As QEC codes improve and become more efficient, they will be essential for building large-scale quantum computers.
Here is an example of how QEC can be used to tackle noise in a quantum communication system:
The sender encodes a single logical qubit into multiple physical qubits using a QEC code.
The sender transmits the physical qubits to the receiver.
The receiver uses the QEC code to detect and correct any errors that occurred during transmission.
The receiver decodes the physical qubits to recover the logical qubit.
If a small number of errors occur during transmission, the QEC code will be able to correct them and the receiver will be able to recover the logical qubit accurately. However, if too many errors occur, the QEC code will not be able to correct them and the receiver will not be able to recover the logical qubit accurately.
QEC can also be used to tackle noise in quantum computers. For example, QEC can be used to protect the qubits in a quantum computer from noise caused by faulty quantum gates or environmental interactions.
QEC is a powerful tool for tackling noise in quantum computing and communication. As QEC codes improve and become more efficient, they will play an essential role in building large-scale quantum computers and enabling reliable quantum communication.



How to run the program

Copy and paste the code below in to a python file

Enter your API token in the IBMQ.enable_account('Insert API token here') part

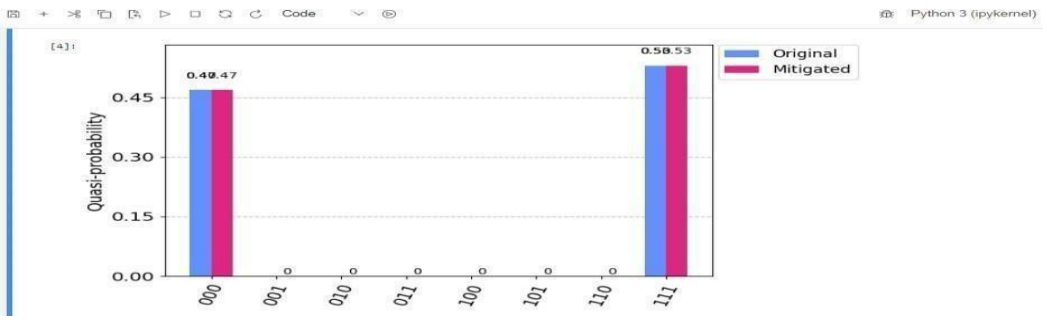
Save and run

Code :

```
!pip install qiskit-ignis


from qiskit import QuantumCircuit, assemble, Aer, transpile
from qiskit.visualization import plot_histogram
from qiskit.ignis.mitigation.import CompleteMeasFitter, complete_meas_cal, tensored_meas_cal

# Define the quantum circuit
qc = QuantumCircuit(3, 3)

# Apply gates and operations to the circuit
qc.h(0)
qc.cx(0, 1)
qc.cx(0, 2)
qc.measure([0, 1, 2], [0, 1, 2])

# Transpile the circuit
backend = Aer.get_backend('qasm_)
transpiled_qc = transpile(qc, backend)

# Simulate the noisy circuit
qobj = assemble(transpiled_qc, shots=1000)
job = backend.run(qobj)
result = job.result()
counts = result.get_counts()

# Perform error mitigation
cal_circuits, state_labels = complete_meas_cal(qubit_list=[, 1, 2])
cal_job = backend.run(assemble(cal_
cal_results = cal_job.result()
meas_fitter = CompleteMeasFitter(cal_
mitigated_counts = meas_fitter.filter.apply(

# Print the original counts
print("Original counts:")
print(counts)

# Print the mitigated counts
print("Mitigated counts:")
print(mitigated_counts)

# Plot the histograms of the original and mitigated counts
```

```
plot_histogram([counts, mitigated_counts], legend=['Original', 'Mitigated'])
```

Output:



Oral Questions:

Q1. How can we adapt error correction codes to different types of noise? How can we design error correction codes that are scalable to large numbers of qubits?
Q2. How can we implement error correction in a way that minimizes overhead?
Q3. How can we combine error correction with other techniques, such as quantum feedback control, to improve the performance of quantum devices?