KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION

-

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

QUANTUM COMPUTING LABORATORY

Subject Code :KG23ACS320

Regulation : KGR23

AcademicYear :2025-2026

III B. TECH II SEMESTER

COMPUTER SCIENCE AND ENGINEERING

KGREDDYCOLLEGEOFENGINEERINGANDTECHNOLOGY

Affiliated by JNTUH,Chilkur,(V),Moinabad(M)R.RDist,TS-501504

KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION

VISION AND MISSION OF THE INSTITUTION

VISION:

To become an institution which is internationally recognized for its holistic approach to engineering, innovative teaching and learning culture, research and entrepreneurial ecosystem, And sustainable social impact in the community.

MISSION:

• To offer undergraduate and post-graduate programs which are supported through industry relevant curriculum and innovative teaching and learning processes that would help students succeed in their professional careers.
• To provide faculty and students with an ecosystem that fosters innovation,research, entrepreneurship, and international exposure through strategic partnerships with government organizations and collaboration with industries.
• To provide holistic learning environment to students which will contribute to their personal and Professional growth and enable them to become leaders in the irrespective fields.
• To contribute to the development of the region by using our technological expertise to work with nearby communities and support them in their social and economic development.

KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VISION:

To be recognized as a department of excellence by stimulating a learning environment in which students and faculty will thriveand grow to achieve their professional,institution a land Societal goals.

MISSION:

• To provide high quality technical education to students that will enable life-long learning and Build expertise in advanced technologies in Computer Science and Engineering.
• To promote research and development by providing opportunities to solve complex engineering problems in collaboration with industry and government agencies.
• To encourage professional development of students that will in culcate ethical values and leadership skills through entrepreneurship while working with the community to address societalissues.

PROGRAMEDUCATIONALOBJECTIVES

PEO1: Graduates will provide solutions to difficult and challenging issues in their profession by applying computer science and engineering theory and principles.
PEO2: Graduate shave successful careers in computer science and engineering fields or will be able to successfully pursue advanced degrees.
PEO3: Graduates will communicate effectively ,work collaboratively and exhibit high levels of professionalism, moral and ethical responsibility.
PEO4: Graduates will develop the ability to understand and analyze engineering issues in a broader perspective with ethical responsibility towards sustainable development.

KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PROGRAM OUTCOMES

POI: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
POII: Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems, reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
POIII: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet specified needs with appropriate consideration for public health and safety, and cultural, societal, and environmental factors.
PO IV: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of information to provide valid conclusions.
POV: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools, including prediction and modeling, to complex engineering activities with an understanding of limitations. .
POVI: The engineer and society: Apply reasoning informed by contextual knowledge to assess societal, health, safety, legal, and cultural issues and the responsibilities relevant to professional engineering practice. .
POVII: Environment and sustainability: Understand the impact of professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of and need for sustainable development.
POVIII: Ethics: Apply ethical principles and commit to professional ethics, responsibilities, and norms of engineering practice.
POIX: Individual and teamwork: Function effectively as an individual, and as a member or leader in diverse teams and multidisciplinary settings.
POX: Communication: Communicate effectively on complex engineering activities with the engineering community and society at large, including writing reports, preparing design documentation, making presentations, and giving and receiving clear instructions. .
POXI: Project management and finance: Demonstrate knowledge and understanding of engineering and management principles and apply these as a member and leader in a team to manage projects in multidisciplinary environments.
PO XII: Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the context of technological change

KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PROGRAM SPECIFIC OUTCOMES

PSO1: Computer Science and Engineering graduates will be able to analyze, design, develop, and test intelligent systems using mathematical foundations and management principles, and apply computational solutions to develop secure applications and hardware prototypes.
PSO2: Graduates will be able to analyze contemporary research issues in various areas of Computer Science and Engineering, identify research gaps, and conduct research in specialized or emerging fields.
PSO3: Graduates will develop skills to solve problems in programming concepts, evaluate environmental and social issues with ethical responsibility, and manage projects in multidisciplinary fields, fostering careers, entrepreneurship, and higher studies.

QUANTUM COMPUTING LABORATORY

Course Code: KG23ACS320 **LTPC**

B.Tech. III Year II Sem **0 0 2 1**

Course Objectives: The objectives of this course for the students are to:

1. Understand Quantum Computing Fundamentals
2. Implement Quantum Algorithms on Simulators
3. Operate Real Quantum Hardware
4. Analyze Quantum Circuit Performance
5. Foster Research and Innovation in Quantum Computing

Course Outcomes:

CO1: Understand principles of quantum computation and basic quantum gates through hands-on simulations. (K2)

CO2: Design and simulate quantum circuits using platforms like Qiskit, Cirq, or Q#. (K6)

CO3: Implement quantum algorithms (e.g., Deutsch–Jozsa, Grover's) on simulators and real quantum hardware. (K3)

CO4: Evaluate and analyze performance of quantum circuits considering noise, gate depth, and errors. (K5)

CO5: Work in teams to design and simulate a simple quantum computing project. (K5)

List of Experiments

S.No	Name ofThe Experiment
1	Python Basics
2	Navigation on Circuit Composer and Qiskit in Quantum Lab
3	Project Preparation – Phase 1 (Analysis of problem statement related to quantum computing)
4	Quantum Hardware and Simulators
5	Implement Single and Multiple Qubit Gates
6	Project Preparation – Phase 2 (Design of the project based on problem statement)
7	Quantum Circuits
8	Visualization Tools (State Vector and Q-Sphere)
9	Project Preparation – Phase 3 (Implementation in cloud environment)
10	Quantum Teleportation in Qiskit
11	Implementation of Grover's Algorithm
12	Project Preparation – Phase 4 (Testing of the software implemented)
13	Project Presentation and Demo (Use case developed
14	Project Thesis Preparation
15	Project Report Submission

TEXTBOOK:

1. Computing with Quantum Cats: From Colossus to Qubits – John Gribbin
2. Quantum Computing From The Ground Up – Riley Tipton Perry

REFERENCE BOOKS:

1. Quantum Computation and Quantum Information – Michael A. Nielsen & Isaac L. Chuang
2. Classical and Quantum Computation – Kitaev, Shen, and Vyalyi
3. Quantum Computing for Computer Scientists – Noson S. Yanofsky & Mirco A. Mannucci

.

EXPERIMENT-1: Python Basics

PROGRAM:

1. Variables and Data Types

Used to store and manipulate data.

```
x = 10          # Integer
y = 3.14        # Float
name = "Qubit"  # String
flag = True     # Boolean
```

In quantum computing, you often store:

- number of qubits
- measurement results
- circuit outputs

2. Input and Output

```
name = input("Enter your name: ")
print("Hello", name)
```

Useful for interactive simulations.

3. Operators

```
a = 5
b = 2

print(a + b)  # Addition
print(a ** b) # Power
print(a % b)  # Modulus
```

Used in probability calculations and logic building.

4. Conditional Statements

```
x = 10

if x > 5:
    print("Greater than 5")
else:
    print("Less or equal to 5")
```

Helps in decision-making during simulations.

5. Loops

For Loop

```
for i in range(5):
    print(i)
```

While Loop

```
i = 0
while i < 5:
    print(i)
    i += 1
```

Used for:

- running multiple simulations
- iterating qubits or gates

6. Functions

```
def add(a, b):
    return a + b
print(add(2, 3))
```

Important for:

- modular quantum programs
- reusable logic

7. Lists (Very Important)

```
qubits = [0, 1, 0]
print(qubits[1])
```

Used to represent:

- qubit states
- measurement results

8. Dictionaries

```
result = {"0": 500, "1": 524}
print(result["0"])
```

In quantum computing:

- stores measurement counts from circuits

9. Libraries (Key for Quantum)

Importing Libraries

```
import math
import numpy as np
```

Important libraries:

- numpy → matrix operations (used in quantum states)
- math → calculations

10. Basic Matrix Handling (Important for Quantum)

```
import numpy as np
matrix = np.array([[1, 0], [0, 1]])
print(matrix)
```

Quantum computing heavily depends on:

- matrices
- vectors
- linear algebra

11. Simple Example (Quantum Style Thinking)

```
import random
qubit = random.choice([0, 1])
print("Measured state:", qubit)
```

Simulates a simple probabilistic measurement.

12. Why Python is Used in Quantum Computing

- Easy to learn and use
- Strong libraries (NumPy, SciPy)
- Works with frameworks like Qiskit
- Supports simulation + real quantum hardware

EXPERIMENT-2: Navigation on Circuit Composer and Qiskit in Quantum Lab.

PROGRAM:

```
from qiskit import QuantumCircuit, Aer, execute
from qiskit.visualization import plot_histogram
import matplotlib.pyplot as plt
# Step 1: Create Quantum Circuit
qc = QuantumCircuit(1, 1)
# Step 2: Apply Hadamard Gate (Superposition)
qc.h(0)
# Step 3: Measure the qubit
qc.measure(0, 0)
# Step 4: Run on Simulator
simulator = Aer.get_backend('qasm_simulator')
result = execute(qc, simulator, shots=1000).result()
# Step 5: Get Output
counts = result.get_counts()
# Step 6: Display Output
print("Measurement Results:", counts)
# Step 7: Plot Histogram
plot_histogram(counts)
plt.show()
```

OUTPUT:

Measurement Results: {'0': 495, '1': 505}

EXPERIMENT-3: Project Preparation – Phase 1(Analysis of problem statement related to quantum computing)

PROGRAM:

Tools Required:

Python

Qiskit

IBM Quantum Experience

Problem Statement:

Design a quantum-based solution to search an element efficiently from an unsorted database using Grover's Algorithm.

Problem Analysis:

Classical Approach:

In classical computing, searching an element in an unsorted database requires O(n) time. In the worst case, all elements must be checked.

Quantum Approach:

Quantum computing uses superposition and parallelism. Grover's Algorithm reduces the search complexity to $O(\sqrt{n})$, making it faster than classical methods.

Key Quantum Concepts Used:

Qubits represent data in quantum systems.

Superposition allows a qubit to exist in multiple states at once.

Entanglement creates correlation between qubits.

Measurement converts the quantum state into classical output.

Required Components:

Quantum circuit

Oracle (to mark the correct solution)

Diffusion operator (to increase probability of correct result)

Input and Output:

Input: Number of qubits and target element

Output: Measured result showing the target element

Feasibility Study:

The problem can be implemented using Qiskit.

It can run on simulators or real quantum hardware.

The complexity is reduced compared to classical methods.

Real hardware may introduce noise and errors.

Expected Outcome:

The probability of measuring the correct element will be higher than others, showing efficiency of the algorithm.

Applications:

Database search

Cryptography

Optimization problems

EXPERIMENT-4: Quantum Hardware and Simulators

PROGRAM:

Tools Required:
Python
Qiskit
IBM Quantum Experience

Theory:

Quantum computers use qubits to perform computations based on quantum mechanics principles such as superposition and entanglement. Quantum circuits can be executed either on simulators or real quantum hardware.

Quantum Hardware:
Quantum hardware refers to real quantum computers that use physical qubits. These systems are developed using technologies like superconducting circuits and trapped ions. Real quantum devices are affected by noise, decoherence, and gate errors.

Simulators:
Simulators are classical programs that imitate quantum systems. They are used for testing and learning because they provide ideal, noise-free results. Simulators are faster and easier to access compared to real hardware.

Types of Simulators in Qiskit:

1. Statevector Simulator
   Displays the complete quantum state (amplitudes of all states).
2. QASM Simulator
   Simulates measurement results with probabilities.
3. Unitary Simulator
   Shows the matrix representation of the circuit.

Program:

```
from qiskit import QuantumCircuit, Aer, execute

# Create a quantum circuit with 1 qubit
qc = QuantumCircuit(1, 1)

# Apply Hadamard gate
qc.h(0)

# Measure the qubit
qc.measure(0, 0)

# Run on QASM simulator
simulator = Aer.get_backend('qasm_simulator')
result = execute(qc, simulator, shots=1000).result()

# Get output
counts = result.get_counts()
print("Simulator Output:", counts)
```

Procedure:

1. Open IBM Quantum Experience
2. Go to Quantum Lab
3. Create a new notebook
4. Enter the program
5. Run the code using simulator
6. Observe the output

Output:

Example output from simulator:

Simulator Output: {'0': 500, '1': 500}

EXPERIMENT-5: Implement Single and Multiple Qubit Gates

PROGRAM:

Tools Required:
Python
Qiskit
IBM Quantum Experience

Single Qubit Gates:
These gates operate on one qubit.
Examples:

- X Gate (NOT gate) – flips the qubit
- H Gate (Hadamard) – creates superposition
- Z Gate – changes phase

Multiple Qubit Gates:
These gates operate on more than one qubit.
Example:

- CNOT Gate (Controlled-NOT) – flips target qubit based on control qubit

Program:

```
from qiskit import QuantumCircuit, Aer, execute

# Create circuit with 2 qubits and 2 classical bits
qc = QuantumCircuit(2, 2)

# Single qubit gates
qc.h(0)      # Hadamard on qubit 0
qc.x(1)      # X gate on qubit 1

# Multiple qubit gate
qc.cx(0, 1)  # CNOT gate (control=0, target=1)

# Measure both qubits
qc.measure([0, 1], [0, 1])

# Run on simulator
simulator = Aer.get_backend('qasm_simulator')
result = execute(qc, simulator, shots=1000).result()

# Output
counts = result.get_counts()
print("Output:", counts)
```

Procedure:

1. Open IBM Quantum Experience
2. Go to Quantum Lab
3. Create a new notebook
4. Enter the program
5. Run the code
6. Observe the output

Output:

Output: {'11': 500, '00': 500}

EXPERIMENT-6: Project Preparation – Phase 2(Design of the project based on problem statement)

PROGRAM:

Tools Required:
Python
Qiskit
IBM Quantum Experience

Problem Statement:
Design a quantum-based solution to search an element efficiently from an unsorted database using Grover's Algorithm.

Project Design:

1. **Overview:**
   The project aims to implement Grover's Algorithm using a quantum circuit to efficiently find a target element from an unsorted database.

2. **System Architecture:**

The system consists of the following components:

- Input module (number of qubits and target element)
- Quantum circuit design
- Oracle implementation
- Diffusion operator
- Measurement module
- Output display

3. **Quantum Circuit Design:**

- Initialize qubits in $|0\rangle$ state
- Apply Hadamard gates to create superposition
- Apply Oracle to mark the target state
- Apply Diffusion operator to amplify probability
- Measure all qubits

4. **Algorithm Steps:**

Step 1: Initialize n qubits
Step 2: Apply Hadamard gate to all qubits
Step 3: Define Oracle function to mark the target state
Step 4: Apply Diffusion operator
Step 5: Repeat steps 3 and 4 ($\sqrt{n}$ times)
Step 6: Measure the qubits

5. **Flow of Execution:**

Input → Circuit Initialization → Superposition → Oracle → Diffusion → Measurement → Output

6. **Design Considerations:**

- Number of qubits depends on database size
- Circuit depth should be minimized
- Noise and errors in real hardware
- Accuracy of measurement

7. **Expected Output:**

- The target element should have the highest probability in the output
- Measurement results will show the correct state more frequently

8. **Applications:**

- Database search
- Cryptography
- Optimization problems

EXPERIMENT-7: Quantum Circuits

PROGRAM:

Tools Required:
Python
Qiskit
IBM Quantum Experience

Components of Quantum Circuits:

- Qubits: Basic units of quantum information
- Quantum Gates: Operations applied to qubits (H, X, Z, CNOT, etc.)
- Measurement: Converts quantum state into classical output

Quantum circuits are represented using circuit diagrams and implemented using programming tools like Qiskit.

Program:

```
from qiskit import QuantumCircuit, Aer, execute
# Create a quantum circuit with 2 qubits and 2 classical bits
qc = QuantumCircuit(2, 2)

# Apply Hadamard gate to qubit 0
qc.h(0)

# Apply CNOT gate (entanglement)
qc.cx(0, 1)

# Measure both qubits
qc.measure([0, 1], [0, 1])

# Run on simulator
simulator = Aer.get_backend('qasm_simulator')
result = execute(qc, simulator, shots=1000).result()

# Get output
counts = result.get_counts()
print("Output:", counts)
```

Procedure:

1. Open IBM Quantum Experience
2. Go to Quantum Lab
3. Create a new notebook
4. Enter the program
5. Run the circuit
6. Observe the output

Output:

Output: {'00': 500, '11': 500}

EXPERIMENT-8: Visualization Tools (State Vector and Q-Sphere)

PROGRAM:

Tools Required:
Python
Qiskit
IBM Quantum Experience

State Vector:
It represents the quantum state as a vector of complex amplitudes. Each value corresponds to the probability amplitude of a basis state.

Example:
For 1 qubit → $|\psi\rangle = a|0\rangle + b|1\rangle$

Q-Sphere:
It is a graphical representation of a quantum state on a sphere.

- Each point represents a basis state
- Size of the point indicates probability
- Color indicates phase

Program:

```
from qiskit import QuantumCircuit, Aer, execute
from qiskit.visualization import plot_state_city, plot_bloch_multivector
from qiskit.visualization import plot_state_qsphere
import matplotlib.pyplot as plt

# Create a quantum circuit
qc = QuantumCircuit(1)

# Apply Hadamard gate
qc.h(0)

# Use statevector simulator
simulator = Aer.get_backend('statevector_simulator')
result = execute(qc, simulator).result()

# Get statevector
statevector = result.get_statevector()

print("State Vector:", statevector)

# Visualization
plot_state_qsphere(statevector)
plt.show()
```

Procedure:

1. Open IBM Quantum Experience
2. Go to Quantum Lab
3. Create a new notebook
4. Enter the program
5. Run the code
6. Observe the state vector and Q-Sphere visualization

Output:

State Vector (example):

State Vector: [0.707+0.j, 0.707+0.j]

EXPERIMENT-9: Project Preparation – Phase 3(Implementation in cloud environment)

PROGRAM:

Tools Required:
Python
Qiskit
IBM Quantum Experience

Problem Statement:
Implement a quantum-based solution to search an element in an unsorted database using Grover's Algorithm in a cloud environment.

Program:

```
from qiskit import QuantumCircuit, Aer, execute

# Create a simple 2-qubit Grover-like circuit
qc = QuantumCircuit(2, 2)

# Step 1: Superposition
qc.h([0, 1])

# Step 2: Oracle (mark |11>)
qc.cz(0, 1)

# Step 3: Diffusion Operator
qc.h([0, 1])
qc.x([0, 1])
qc.h(1)
qc.cx(0, 1)
qc.h(1)
qc.x([0, 1])
qc.h([0, 1])

# Step 4: Measurement
qc.measure([0, 1], [0, 1])

# Run on simulator (cloud environment)
simulator = Aer.get_backend('qasm_simulator')
result = execute(qc, simulator, shots=1000).result()

# Output
counts = result.get_counts()
print("Output:", counts)
```

Procedure:

1. Open IBM Quantum Experience
2. Login to your account
3. Navigate to Quantum Lab

4. Create a new notebook
5. Enter the program
6. Run the circuit on simulator
7. (Optional) Select real quantum backend and execute
8. Observe the output

Output:

Example output:

Output: {'11': 800, '00': 100, '01': 50, '10': 50}

EXPERIMENT-10: Quantum Teleportation in Qiskit

PROGRAM:

Tools Required:
Python
Qiskit
IBM Quantum Experience

Working Principle:

1. Create an entangled pair of qubits
2. Combine the unknown qubit with one entangled qubit
3. Perform measurement
4. Apply correction operations on the target qubit
5. The original state is reconstructed on the target qubit

Program:

```
from qiskit import QuantumCircuit, Aer, execute

# Create circuit with 3 qubits and 3 classical bits
qc = QuantumCircuit(3, 3)

# Step 1: Prepare state to teleport (apply H gate)
qc.h(0)

# Step 2: Create entanglement between qubit 1 and 2
qc.h(1)
qc.cx(1, 2)

# Step 3: Bell measurement
qc.cx(0, 1)
qc.h(0)

# Step 4: Measure qubits 0 and 1
qc.measure([0, 1], [0, 1])

# Step 5: Conditional operations
qc.cx(1, 2)
qc.cz(0, 2)

# Step 6: Measure target qubit
qc.measure(2, 2)

# Run on simulator
simulator = Aer.get_backend('qasm_simulator')
result = execute(qc, simulator, shots=1000).result()

# Output
```

```
counts = result.get_counts()
print("Output:", counts)
```

Procedure:

1. Open IBM Quantum Experience
2. Go to Quantum Lab
3. Create a new notebook
4. Enter the program
5. Run the circuit
6. Observe the output

Output:

Example output:

Output: {'000': 250, '011': 250, '101': 250, '110': 250}

EXPERIMENT-11: Implementation of Grover's Algorithm

PROGRAM:

Tools Required:
Python
Qiskit
IBM Quantum Experience

Main Components:

1. Superposition (Hadamard gates)
2. Oracle (marks the target state)
3. Diffusion operator (amplifies probability)

Program:

```
from qiskit import QuantumCircuit, Aer, execute

# Create 2-qubit circuit
qc = QuantumCircuit(2, 2)

# Step 1: Apply Hadamard gates (superposition)
qc.h([0, 1])

# Step 2: Oracle (mark state |11>)
qc.cz(0, 1)

# Step 3: Diffusion operator
qc.h([0, 1])
qc.x([0, 1])
qc.h(1)
qc.cx(0, 1)
qc.h(1)
qc.x([0, 1])
qc.h([0, 1])

# Step 4: Measurement
qc.measure([0, 1], [0, 1])

# Run on simulator
simulator = Aer.get_backend('qasm_simulator')
result = execute(qc, simulator, shots=1000).result()

# Output
counts = result.get_counts()
print("Output:", counts)
```

Procedure:

1. Open IBM Quantum Experience
2. Go to Quantum Lab
3. Create a new notebook
4. Enter the program
5. Run the circuit
6. Observe the output

Output:

Example output:

Output: {'11': 820, '00': 60, '01': 60, '10': 60}

EXPERIMENT-12: Project Preparation – Phase 4(Testing of the software implemented)

PROGRAM:

Tools Required:
Python
Qiskit
IBM Quantum Experience

Problem Statement:
Test the implemented quantum-based solution for searching an element in an unsorted database using Grover's Algorithm.

Testing Methodology:

1. Functional Testing
   Verify whether the circuit correctly identifies the target state.
2. Performance Testing
   Analyze probability distribution of outputs.
3. Simulation Testing
   Run the circuit on a simulator to get ideal results.
4. Hardware Testing
   Execute on real quantum hardware and compare results.

Test Cases:

Test Case 1:
Input: 2 qubits, target state |11⟩
Expected Output: Highest probability for |11⟩

Test Case 2:
Input: Different number of shots (e.g., 500, 1000)
Expected Output: Similar probability distribution

Test Case 3:
Input: Execution on real hardware
Expected Output: Slight variation due to noise

Program (Testing):

```
from qiskit import QuantumCircuit, Aer, execute

# Grover's Algorithm Test Circuit
qc = QuantumCircuit(2, 2)

qc.h([0, 1])
qc.cz(0, 1)

qc.h([0, 1])
qc.x([0, 1])
qc.h(1)
```

```
qc.cx(0, 1)
qc.h(1)
qc.x([0, 1])
qc.h([0, 1])

qc.measure([0, 1], [0, 1])

simulator = Aer.get_backend('qasm_simulator')
result = execute(qc, simulator, shots=1000).result()

counts = result.get_counts()
print("Test Output:", counts)
```

Output:

Example output:

Test Output: {'11': 830, '00': 60, '01': 55, '10': 55}

EXPERIMENT-13: Project Presentation and Demo (Use case developed)

PROGRAM:

Tools Required:
Python
Qiskit
IBM Quantum Experience
Presentation tools (PowerPoint or similar)

Use Case:
Efficient search in an unsorted database using Grover's Algorithm.

Introduction:

The project demonstrates how quantum computing can be used to improve search efficiency. In classical systems, searching an unsorted database requires linear time. Using quantum computing, Grover's Algorithm reduces the search time significantly.

Problem Statement:

To design and implement a quantum algorithm that efficiently finds a target element in an unsorted database.

Solution Overview:

The solution uses Grover's Algorithm implemented through a quantum circuit. The algorithm uses superposition, oracle marking, and amplitude amplification to identify the target state.

System Architecture:

Input → Quantum Circuit → Oracle → Diffusion Operator → Measurement → Output

Demonstration Steps:

1. Initialize the quantum circuit with required qubits
2. Apply Hadamard gates to create superposition
3. Implement oracle to mark the target state
4. Apply diffusion operator to amplify probability
5. Measure the qubits
6. Display output results

Demo Program:

```
from qiskit import QuantumCircuit, Aer, execute

qc = QuantumCircuit(2, 2)

qc.h([0, 1])
qc.cz(0, 1)

qc.h([0, 1])
```

```
qc.x([0, 1])
qc.h(1)
qc.cx(0, 1)
qc.h(1)
qc.x([0, 1])
qc.h([0, 1])

qc.measure([0, 1], [0, 1])

simulator = Aer.get_backend('qasm_simulator')
result = execute(qc, simulator, shots=1000).result()

counts = result.get_counts()
print("Demo Output:", counts)
```

Output:

Example output:

Demo Output: {'11': 820, '00': 60, '01': 60, '10': 60}

EXPERIMENT-14: Project Thesis Preparation

PROGRAM:

Tools Required:
Python
Qiskit
IBM Quantum Experience
Word Processor (MS Word or similar)

Structure of Project Thesis:

1. **Title Page**
   - Project Title
   - Student Name(s)
   - Course Name and Code
   - Institution Name
   - Guide Name

2. **Abstract**
   A brief summary of the project including objective, method, and results.

3. **Introduction**
   - Overview of quantum computing
   - Importance of the problem
   - Objective of the project

4. **Literature Survey**
   - Previous work related to quantum search algorithms
   - Overview of Grover's Algorithm
   - Tools and technologies used

5. **Problem Statement**
   Clearly define the problem being solved.

6. **Methodology**
   - Quantum circuit design
   - Explanation of algorithm steps
   - Tools used (Qiskit, simulators)

7. **System Design**
   - Architecture diagram
   - Flow of execution
   - Circuit diagrams

8. **Implementation**
   - Program code
   - Explanation of code
   - Screenshots of execution

9. **Results and Analysis**

- Output results
- Graphs (histograms)
- Comparison between simulator and hardware

10. **Applications**

- Real-world uses of the project

11. **Conclusion**

- Summary of work
- Achievements
- Observations

12. **Future Scope**

- Improvements
- Advanced applications

13. **References**

- Books
- Research papers
- Online resources

Sample Abstract:

This project presents the implementation of Grover's Algorithm using Qiskit for efficient searching in an unsorted database. The algorithm utilizes quantum principles such as superposition and amplitude amplification to reduce search complexity. The results demonstrate improved performance compared to classical methods.

OUTPUT:
The project thesis document was successfully prepared with all required sections.

EXPERIMENT-15: Project Report Submission

PROGRAM:

Tools Required:
Python
Qiskit
IBM Quantum Experience
Word Processor (MS Word or similar)

Structure of Project Thesis:

1. **Title Page**
   - Project Title
   - Student Name(s)
   - Course Name and Code
   - Institution Name
   - Guide Name

2. **Abstract**
   A brief summary of the project including objective, method, and results.

3. **Introduction**
   - Overview of quantum computing
   - Importance of the problem
   - Objective of the project

4. **Literature Survey**
   - Previous work related to quantum search algorithms
   - Overview of Grover's Algorithm
   - Tools and technologies used

5. **Problem Statement**
   Clearly define the problem being solved.

6. **Methodology**
   - Quantum circuit design
   - Explanation of algorithm steps
   - Tools used (Qiskit, simulators)

7. **System Design**
   - Architecture diagram
   - Flow of execution
   - Circuit diagrams

8. **Implementation**
   - Program code
   - Explanation of code
   - Screenshots of execution

9. **Results and Analysis**

- Output results
- Graphs (histograms)
- Comparison between simulator and hardware

10. **Applications**

- Real-world uses of the project

11. **Conclusion**

- Summary of work
- Achievements
- Observations

12. **Future Scope**

- Improvements
- Advanced applications

13. **References**

- Books
- Research papers
- Online resources

Sample Abstract:

This project presents the implementation of Grover's Algorithm using Qiskit for efficient searching in an unsorted database. The algorithm utilizes quantum principles such as superposition and amplitude amplification to reduce search complexity. The results demonstrate improved performance compared to classical methods.

OUTPUT:
The project thesis document was successfully prepared with all required sections.