PRESIDENCY UNIVERSITY

GAIN MORE KNOWLEDGE
REACH GREATER HEIGHTS

CSE 3080: QUANTUM COMPUTING

MINI PROJECT

SECTION: 6-CAI3

20201CAI0156 - ARYA KUMAR JENA
20201CAI0179 - ENOSHA P
20201CAI0195 - DARNESH D P
20201CAI0208 - SANOOP P NAMBIAR
20201CAI0215 - V VASANTH

Task:

Create a program that builds an oracle for a given string (e.g. given 01101, will return a QuantumCircuit that inverts the phase of the state **|01101>** and leaves all other states unchanged.

```
# Importing standard Qiskit libraries
from qiskit import QuantumCircuit, transpile
from qiskit.tools.jupyter import *
from qiskit.visualization import *
from ibm_quantum_widgets import *
from qiskit_aer import AerSimulator

# qiskit-ibmq-provider has been deprecated.
# Please see the Migration Guides in
https://ibm.biz/provider_migration_guide for more detail.
from qiskit_ibm_runtime import QiskitRuntimeService, Sampler, Estimator,
Session, Options

# Loading your IBM Quantum account(s)
service = QiskitRuntimeService(channel="ibm_quantum")

# Invoke a primitive inside a session. For more details see
https://qiskit.org/documentation/partners/qiskit_ibm_runtime/tutorials.html
# with Session(backend=service.backend("ibmq_qasm_simulator")):
#     result = Sampler().run(circuits).result()


from qiskit import QuantumCircuit, QuantumRegister, Aer, execute

def build_oracle(input_string):
    num_qubits = len(input_string)

    qr = QuantumRegister(num_qubits)
    qc = QuantumCircuit(qr)

    # Apply X gates to qubits corresponding to 1s in the input string
    for i, bit in enumerate(input_string):
        if bit == 1:
            qc.x(qr[i])

    # Apply the phase inversion to the state |01101>
    qc.cz(qr[0], qr[num_qubits-1])

    # Apply X gates again to qubits corresponding to 1s in the input string
    for i, bit in enumerate(input_string):
        if bit == 1:
            qc.x(qr[i])

    return qc
```

```python
input_string = [0, 1, 1, 0, 1]
oracle_circuit = build_oracle(input_string)

simulator = Aer.get_backend('statevector_simulator')
job = execute(oracle_circuit, simulator)
result = job.result()
statevector = result.get_statevector()

# Print the state before and after applying the oracle
print("State before the oracle:")
print(statevector)

print("\nState after the oracle:")
print(statevector.conj())

# Draw the quantum circuit
oracle_circuit.draw()
```

```
State before the oracle:
Statevector([ 1.+0.j, -0.+0.j,  0.+0.j, -0.+0.j,  0.+0.j, -0.+0.j,  0.+0.j,
             -0.+0.j,  0.+0.j, -0.+0.j,  0.+0.j, -0.+0.j,  0.+0.j, -0.+0.j,
              0.+0.j, -0.+0.j,  0.+0.j,  0.+0.j,  0.+0.j,  0.+0.j,  0.+0.j,
              0.+0.j,  0.+0.j,  0.+0.j,  0.+0.j,  0.+0.j,  0.+0.j,  0.+0.j,
              0.+0.j,  0.+0.j,  0.+0.j,  0.+0.j],
            dims=(2, 2, 2, 2, 2))

State after the oracle:
[ 1.-0.j -0.-0.j  0.-0.j -0.-0.j  0.-0.j -0.-0.j  0.-0.j -0.-0.j  0.-0.j
 -0.-0.j  0.-0.j -0.-0.j  0.-0.j -0.-0.j  0.-0.j -0.-0.j  0.-0.j  0.-0.j
  0.-0.j  0.-0.j  0.-0.j  0.-0.j  0.-0.j  0.-0.j  0.-0.j  0.-0.j  0.-0.j
  0.-0.j  0.-0.j  0.-0.j  0.-0.j  0.-0.j]
/tmp/ipykernel_328/501170095.py:37: DeprecationWarning: The return type of saved statevector
  print(statevector.conj())
```