Assignment 3

Problem Statement:

Implementing a 5 qubit Quantum Fourier Transform

Objective:

1. Understand Quantum Fourier Transform
2. Create 5 qubit Quantum Fourier Transform

Outcome

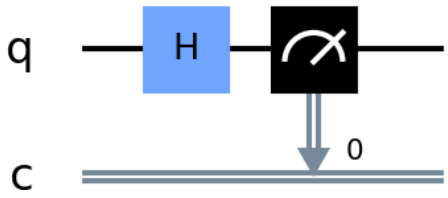
Displays circuit for 5 qubit Quantum Fourier Transform.

Theory:

What is the Quantum Fourier Transform?
The Quantum Fourier Transform (QFT) is a circuit that transforms the state of the qubit from the computational basis to the Fourier basis. *Note that the Fourier basis is just another term for the Hadamard basis.* As such the easiest way to implement a QFT is with Hadamard gates and Controlled U1 gates.

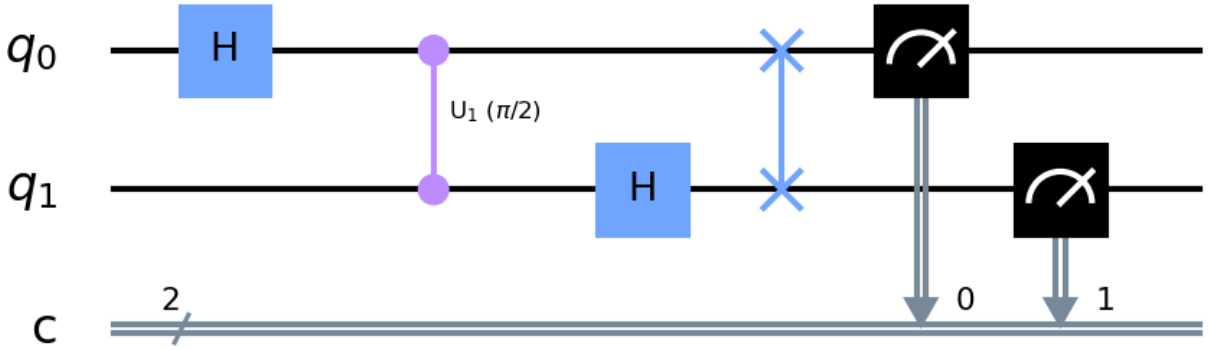
Note: A Controlled U1 gate is just a gate that implements a single rotation around the Z-axis (phase) of the target qubit if the control qubit is 1.

Circuit diagram of a 1 qubit QFT



The simplest QFT is a 1 qubit QFT which just implements a Hadamard gate.

However if we implement a 2 qubit QFT then you can see how the controlled U1 are used:



2 qubit QFT

First we implement a Hadamard gate which puts q0 in to superposition. Next we apply a controlled U1 gate with a rotation of pi/2 to q1. After this a Hadamard gate is applied to q1. Next we apply a swap gate to q0 and q1.

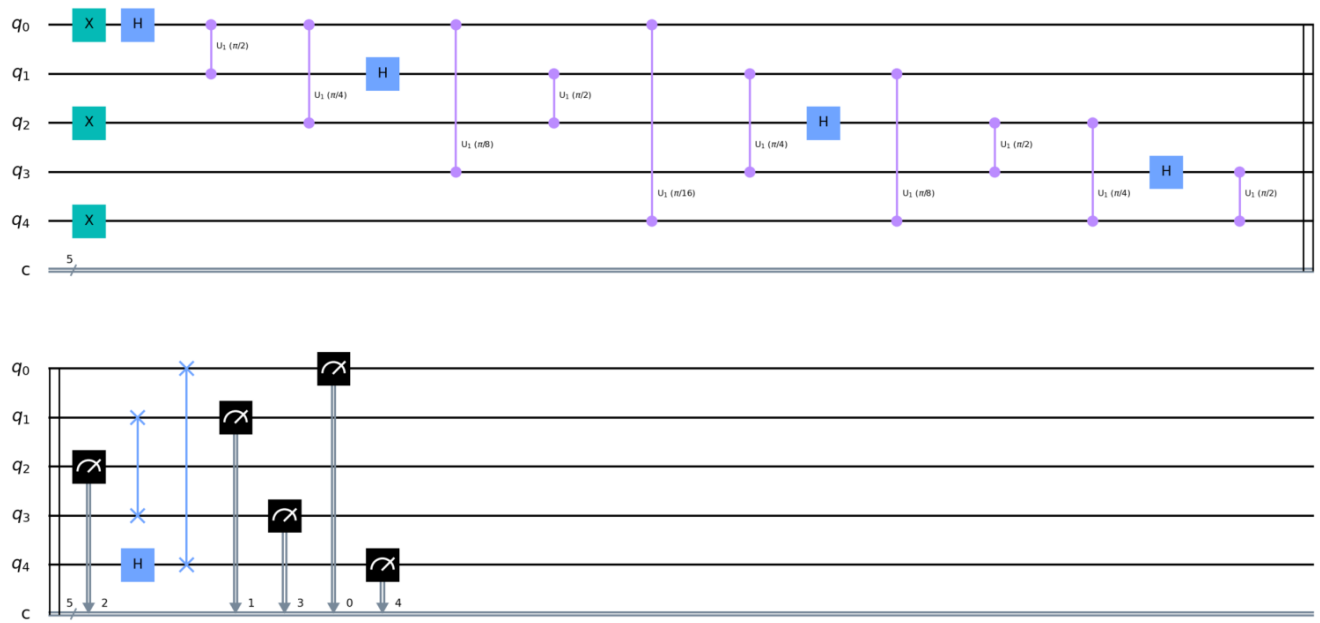
Note that these swap gates are not needed if the QFT is implemented at the end of your circuit.

After this both qubits will be in superposition but whatever computational value (1 or 0) will be encoded in to the Hadamard basis of the qubit.

To encode values on N qubits we have to double the rotation value of each qubit. For example the diagram below shows a 5 qubit QFT.

Notice how for q0 it applies a rotation of pi/2 for q1 then pi/4 for q2 then pi/8 for q3 and so on. This pattern repeats for each qubit. When all rotations have been applied to a qubit it is put in to superposition using a Hadamard gate. Then it can be used as a control qubit to apply rotations to target qubits below it.

Implementation



Circuit diagram of a 5 qubit QFT

Implementing a 5 qubit Quantum Fourier Transform in qiskit
In qiskit we could implement the 5 qubit QFT by implementing all the gates in the diagram above.

In qiskit you can use the QFT() function as follows:

QFT(num_qubits=None, approximation_degree=0, do_swaps=True, inverse=False, insert_barriers=False, name='qft')
Where:

num_qubits: The number of qubits we want to add to the QFT (in our case it is 5)

approximation_degree: This allows us to reduce circuit depth by ignoring phase rotations under a certain value

do_swaps: If set to true then we use swap gates in the QFT

inverse: If set to true we implement the inverse QFT

insert barrier: If set to true then we insert barriers

For example in our 5 qubit QFT we implement the following:

QFT(num_qubits=5, approximation_degree=0, do_swaps=True, inverse=False, insert_barriers=True, name='qft')

If we encode 1010 on to a QFT and then measure it we will get random values since the qubits have been put in to superposition and the values we encoded in to the computational basis are now encoded in the Hadamard basis of each qubit via the controlled U1 gates.

Inverse Quantum Fourier Transform
To get our values back we can use the inverse QFT. This reverses all the rotations done in the QFT above.

For example is there was a rotation of Pi in the QFT then the inverse QFT will do a rotation of -Pi.

In qiskit we can get the values back by implementing an inverse QFT by setting inverse to true.

For example:

QFT(num_qubits=5, approximation_degree=0, do_swaps=True, inverse=True, insert_barriers=True, name='qft')

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

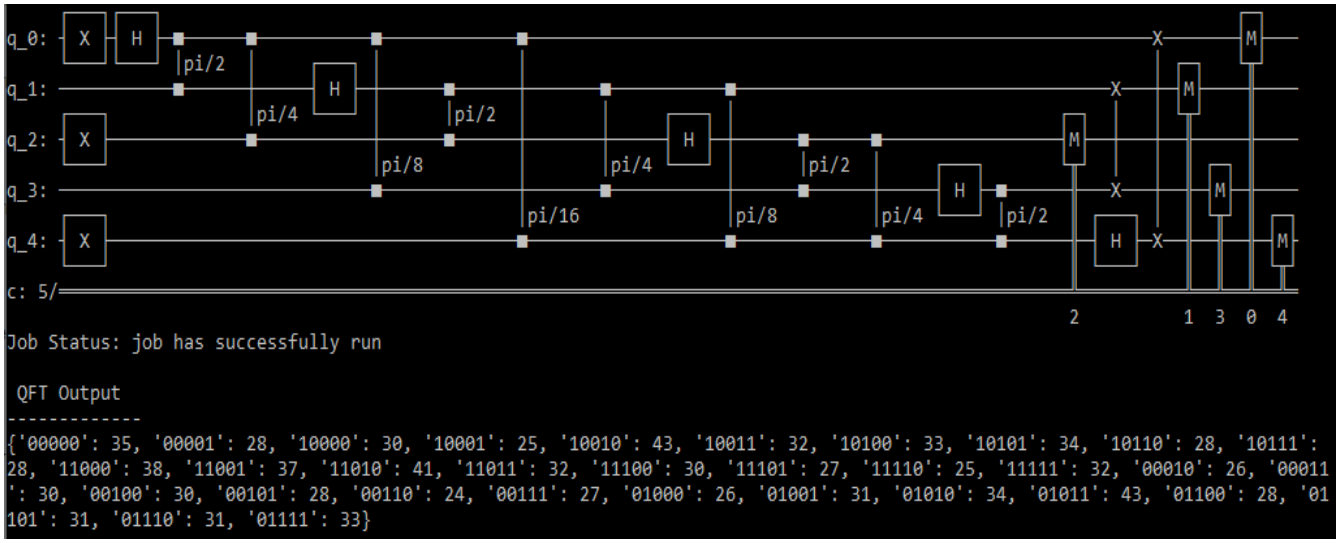
Code :

```
from qiskit import QuantumRegister, ClassicalRegister
from qiskit import QuantumCircuit, execute,IBMQ
from qiskit.tools.monitor import job_monitor
from qiskit.circuit.library import QFT
import numpy as np

pi = np.pi

IBMQ.enable_account(‘ENTER API KEY HERE’)
provider = IBMQ.get_provider(hub='ibm-q')

backend = provider.get_backend('ibmq_qasm_simulator')

q = QuantumRegister(5,'q')
c = ClassicalRegister(5,'c')
```

```
circuit = QuantumCircuit(q,c)

circuit.x(q[4])
circuit.x(q[2])
circuit.x(q[0])
circuit += QFT(num_qubits=5, approximation_degree=0, do_swaps=True, inverse=False,
insert_barriers=False, name='qft')
circuit.measure(q,c)
circuit.draw(output='mpl', filename='qft1.png')
print(circuit)

job = execute(circuit, backend, shots=1000)

job_monitor(job)

counts = job.result().get_counts()

print("\n QFT Output")
print(" ------------ ")
print(counts)
input()
```

Output

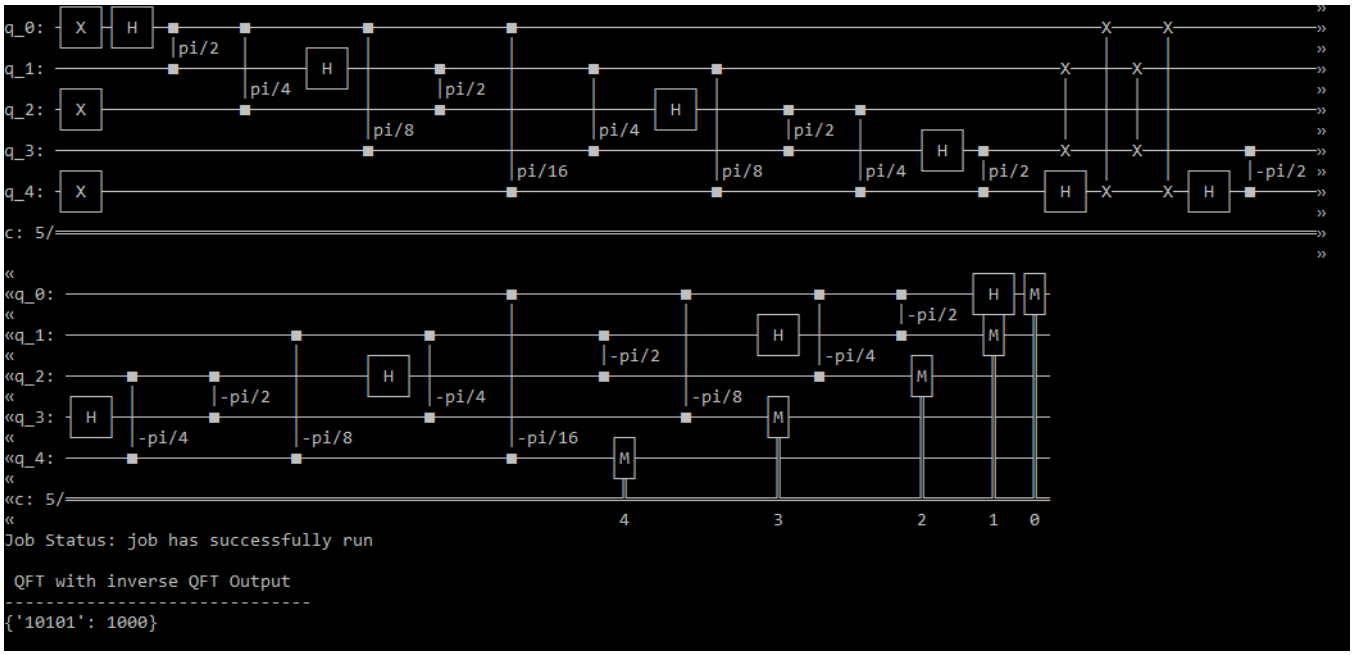
Here is the output when running the QFT:



Output when running the 5 qubit QFT. Notice how we get multiple values back since the qubits are in superposition and the value is encoded in the Hadamard basis.

Here is the output when running the QFT with the inverse QFT.

Oral Questions:



Notice how we get the 1010 back!