Quantum Phase Estimation with Q

For a unitary gate such as T-gate, where,

$$T = \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{4}} \end{bmatrix}$$

Quantum Phase Estimation would give us,

$$T|\psi\rangle = \alpha|0\rangle + e^{2i\pi\theta}\beta|1\rangle$$

where, $\theta = ?$

Your Task

Your task is to construct a Quantum Phase Estimation algorithm using 4 qubits, for estimating the phase of a T-gate, as given above.

In [1]:
```
operation initialize (q: Qubit []) : Unit {
    // WRITE YOUR CODE HERE
    H(q[0]);
    H(q[1]);
    H(q[2]);
    X(q[3]);
}
```

Out[1]:
- initialize

In [2]:
```
operation add_rotations(q: Qubit[], num_qubits: Int): Unit {
    // WRITE YOUR CODE HERE
    // Use R1Frac for implementing controlled rotations.
    mutable repetitions = 1;
    for i in 0 .. num_qubits-2{
        for j in 0 .. repetitions-1{
            //mutable rot = R1Frac(1,2,q[num_qubits-1]);
            //Controlled rot([q[i]],num_qubits-1);
            (Controlled R1Frac)([q[i]],(1,2,q[num_qubits-1]));
        }
        set repetitions = repetitions*2;
    }

    // Swap
    // Implement Swap with the help of Microsoft.Quantum.Canon.SwapReverseReg
    SwapReverseRegister([q[0],q[2]]);
}
```

Out[2]:
- add_rotations

In [3]:
```
operation QFT(q: Qubit[], num_qubits: Int): Unit is Adj + Ctl {
    // WRITE YOUR CODE HERE
    // Use R1Frac for implementing controlled rotations.
    (Controlled R1Frac)([q[1]],(-1,1,q[2]));
    (Controlled R1Frac)([q[0]],(-1,2,q[2]));
    H(q[1]);
    (Controlled R1Frac)([q[0]],(-1,1,q[1]));
    H(q[0]);
}
```

Out[3]:

- QFT

At the end, you can choose to perform measurement and reset all the qubits using %simulate (or DumpMachine), or draw the circuit using %trace.

Don't forget to change the return type of the operation in case you perform measurement.

In [4]:
```
// Call all of the above functions in a single operation to build the QPE cir

operation QPE (): Unit {
    use q = Qubit[4];
    let len = Length(q);

    // Initialize the circuit
    initialize(q);

    // Add rotations wrt T gate.
    add_rotations(q, 4);

    // Implement QFT-dagger (adjoint)
    Adjoint QFT(q, 4);

    // R
    ResetAll(q);
}
```

Out[4]:

- QPE

In [5]:
```
%trace QPE
```

In [6]:
```
%simulate QPE
```

Out[6]: ()

Resources

You can refer to the following link for a better look at the algorithm implementation: Qiskit Implementation of QPE