Distinguish Unitaries Kata

The **Distinguish Unitaries** quantum kata offers tasks in which you are given a unitary and have to figure out which of the list it is by designing and performing experiments on it.

Each task is wrapped in one operation preceded by the description of the task. Your goal is to fill in the blank (marked with `// ...` comments) with some Q# code that solves the task. To verify your answer, run the cell using Ctrl+Enter (⌘+Enter on macOS).

The tasks are given in approximate order of increasing difficulty; harder ones are marked with asterisks.

Part I. Single-Qubit Gates

Task 1.1. Identity or Pauli X?

Input: An operation that implements a single-qubit unitary transformation: either the identity (**I** gate) or the Pauli X gate (**X** gate). The operation will have Adjoint and Controlled variants defined.

Output: 0 if the given operation is the **I** gate, 1 if the given operation is the **X** gate.

You are allowed to apply the given operation and its adjoint/controlled variants exactly **once**.

In [ ]:
```
%kata T101_DistinguishIfromX

operation DistinguishIfromX (unitary : (Qubit => Unit is Adj+Ctl)) : Int {
    // ...
    return -1;
}
```

Can't come up with a solution? See the explained solution in the Distinguish Unitaries Workbook.

Task 1.2. Identity or Pauli Z?

Input: An operation that implements a single-qubit unitary transformation: either the identity (**I** gate) or the Pauli Z gate (**Z** gate). The operation will have Adjoint and Controlled variants defined.

Output: 0 if the given operation is the **I** gate, 1 if the given operation is the **Z** gate.

You are allowed to apply the given operation and its adjoint/controlled variants exactly **once**.

In [ ]:
```
%kata T102_DistinguishIfromZ

operation DistinguishIfromZ (unitary : (Qubit => Unit is Adj+Ctl)) : Int {
    // ...
    return -1;
}
```

Can't come up with a solution? See the explained solution in the Distinguish Unitaries Workbook.

Task 1.3. Z or S?

Input: An operation that implements a single-qubit unitary transformation: either the **Z** gate or the **S** gate. The operation will have Adjoint and Controlled variants defined.

Output: 0 if the given operation is the **Z** gate, 1 if the given operation is the **S** gate.

You are allowed to apply the given operation and its adjoint/controlled variants at most **twice**.

In [ ]:
```
%kata T103_DistinguishZfromS

operation DistinguishZfromS (unitary : (Qubit => Unit is Adj+Ctl)) : Int {
    // ...
    return -1;
}
```

Can't come up with a solution? See the explained solution in the Distinguish Unitaries Workbook.

Task 1.4. Hadamard or X?

Input: An operation that implements a single-qubit unitary transformation: either the Hadamard (**H**) gate or the **X** gate. The operation will have Adjoint and Controlled variants defined.

Output: 0 if the given operation is the **H** gate, 1 if the given operation is the **X** gate.

You are allowed to apply the given operation and its adjoint/controlled variants at most **twice**.

In [3]:
```
%kata T104_DistinguishHfromX

operation DistinguishHfromX (unitary : (Qubit => Unit is Adj+Ctl)) : Int {
    use q = Qubit();
    within{
        unitary(q);
    }
    apply{
        X(q);
    }

    if(M(q) == Zero){
        return 0;
    }
    else{
        return 1;
    }
}
```

Out[3]:
```
Success!
```

Can't come up with a solution? See the explained solution in the Distinguish Unitaries Workbook.

Task 1.5. Z or −Z?

Input: An operation that implements a single-qubit unitary transformation: either the **Z** gate or the minus **Z** gate (i.e., the gate $-|0\rangle\langle 0| + |1\rangle\langle 1|$). The operation will have Adjoint and Controlled variants defined.

Output: 0 if the given operation is the **Z** gate, 1 if the given operation is the $-$**Z** gate.

You are allowed to apply the given operation and its adjoint/controlled variants exactly **once**.

In [15]:
```
%kata T105_DistinguishZfromMinusZ

operation DistinguishZfromMinusZ (unitary : (Qubit => Unit is Adj+Ctl)) : Int
    use q = Qubit[2];
    H(q[0]);
    Controlled unitary([q[0]],q[1]);

    H(q[0]);
    if (M(q[0]) == Zero){
        return 0;
    }
    else{
        return 1;
    }
}
```

Out[15]:
```
Success!
```

Can't come up with a solution? See the explained solution in the Distinguish Unitaries Workbook.

Task 1.6. Rz or R1 (arbitrary angle)?

Input: An operation that implements a single-qubit unitary transformation: either the **Rz** gate or the **R1** gate.

This operation will take two parameters: the first parameter is the rotation angle, in radians, and the second parameter is the qubit to which the gate should be applied (matching normal **Rz** and **R1** gates in Q#). The operation will have Adjoint and Controlled variants defined.

Output: 0 if the given operation is the **Rz** gate, 1 if the given operation is the **R1** gate.

You are allowed to apply the given operation and its adjoint/controlled variants exactly **once**.

In [ ]:
```
%kata T106_DistinguishRzFromR1

operation DistinguishRzFromR1 (unitary : ((Double, Qubit) => Unit is Adj+Ctl)
    // ...
    return -1;
}
```

Can't come up with a solution? See the explained solution in the Distinguish Unitaries Workbook?).

Task 1.7. Y or XZ?

Input: An operation that implements a single-qubit unitary transformation: either the **Y** gate or the sequence of Pauli **Z** and Pauli **X** gates (equivalent to applying the **Z** gate followed by the **X** gate). The operation will have Adjoint and Controlled variants defined.

Output: 0 if the given operation is the **Y** gate, 1 if the given operation is the **XZ** gate.

You are allowed to apply the given operation and its adjoint/controlled variants at most **twice**.

In [ ]:
```
%kata T107_DistinguishYfromXZ

operation DistinguishYfromXZ (unitary : (Qubit => Unit is Adj+Ctl)) : Int {
    // ...
    return -1;
}
```

Can't come up with a solution? See the explained solution in the Distinguish Unitaries Workbook.

Task 1.8. Y, XZ, −Y or −XZ?

Input: An operation that implements a single-qubit unitary transformation: either the **Y** gate (possibly with an extra global phase of -1) or the sequence of Pauli **Z** and Pauli **X** gates (possibly with an extra global phase of -1). The operation will have Adjoint and Controlled variants defined.

Output:

- 0 if the given operation is the **Y** gate,
- 1 if the given operation is the $-$**XZ** gate,
- 2 if the given operation is the $-$**Y** gate,
- 3 if the given operation is the **XZ** gate.

You are allowed to apply the given operation and its adjoint/controlled variants at most **three times**.

In [ ]:
```
%kata T108_DistinguishYfromXZWithPhases

operation DistinguishYfromXZWithPhases (unitary : (Qubit => Unit is Adj+Ctl))
    // ...
    return -1;
}
```

Can't come up with a solution? See the explained solution in the Distinguish Unitaries Workbook.

Task 1.9. Rz or Ry (fixed angle)?

Inputs:

1. An angle $\theta \in [0.01\pi; 0.99\pi]$.
2. An operation that implements a single-qubit unitary transformation: either the $R_z(\theta)$ gate or the $R_y(\theta)$ gate.

The operation will have Adjoint and Controlled variants defined.

Output: 0 if the given operation is the **Rz** gate, 1 if the given operation is the **Ry** gate.

You are allowed to apply the given operation and its adjoint/controlled variants **any number of times**.

In [ ]:
```
%kata T109_DistinguishRzFromRy

operation DistinguishRzFromRy (theta : Double, unitary : (Qubit => Unit is Adj
    // ...
    return -1;
}
```

Can't come up with a solution? See the explained solution in the Distinguish Unitaries Workbook?).

Task 1.10*. Rz or R1 (fixed angle)?

Inputs:

1. An angle $\theta \in [0.01\pi; 0.99\pi]$.
2. An operation that implements a single-qubit unitary transformation: either the $R_z(\theta)$ gate or the $R_1(\theta)$ gate.

The operation will have Adjoint and Controlled variants defined.

Output: 0 if the given operation is the **Rz** gate, 1 if the given operation is the **R1** gate.

You are allowed to apply the given operation and its adjoint/controlled variants **any number of times**.

In [ ]:
```
%kata T110_DistinguishRzFromR1WithAngle

operation DistinguishRzFromR1WithAngle (theta : Double, unitary : (Qubit => U
    // ...
    return -1;
}
```

Task 1.11. Distinguish 4 Pauli unitaries

Input: An operation that implements a single-qubit unitary transformation: either the identity (**I** gate) or one of the Pauli gates (**X**, **Y** or **Z** gate). The operation will have Adjoint and Controlled variants defined.

Output:

- 0 if the given operation is the **I** gate,
- 1 if the given operation is the **X** gate,
- 2 if the given operation is the **Y** gate,
- 3 if the given operation is the **Z** gate.

You are allowed to apply the given operation and its adjoint/controlled variants exactly **once**.

In [ ]:
```
%kata T111_DistinguishPaulis

operation DistinguishPaulis (unitary : (Qubit => Unit is Adj+Ctl)) : Int {
    // ...
    return -1;
}
```

Can't come up with a solution? See the explained solution in the Distinguish Unitaries Workbook.

Part II. Multi-Qubit Gates

Task 2.1. $I \otimes X$ or CNOT?

Input: An operation that implements a two-qubit unitary transformation: either the $I \otimes X$ (the X gate applied to the second qubit) or the CNOT gate with the first qubit as control and the second qubit as target.

- The operation will accept an array of qubits as input, but it will fail if the array is empty or has one or more than two qubits.

- The operation will have Adjoint and Controlled variants defined.

Output: 0 if the given operation is $I \otimes X$, 1 if the given operation is the CNOT gate.

You are allowed to apply the given operation and its adjoint/controlled variants exactly **once**.

In [ ]:
```
%kata T201_DistinguishIXfromCNOT

operation DistinguishIXfromCNOT (unitary : (Qubit[] => Unit is Adj+Ctl)) : Int
    // ...
    return -1;
}
```

Can't come up with a solution? See the explained solution in the Distinguish Unitaries Workbook.

Task 2.2. Figure out the direction of CNOT

Input: An operation that implements a two-qubit unitary transformation: either the CNOT gate with the first qubit as control and the second qubit as target (CNOT_{12}) or the CNOT gate with the second qubit as control and the first qubit as target (CNOT_{21}).

- The operation will accept an array of qubits as input, but it will fail if the array is empty or has one or more than two qubits.
- The operation will have Adjoint and Controlled variants defined.

Output: 0 if the given operation is CNOT_{12}, 1 if the given operation is CNOT_{21}.

You are allowed to apply the given operation and its adjoint/controlled variants exactly **once**.

In [ ]:
```
%kata T202_CNOTDirection

operation CNOTDirection (unitary : (Qubit[] => Unit is Adj+Ctl)) : Int {
    // ...
    return -1;
}
```

Can't come up with a solution? See the explained solution in the Distinguish Unitaries Workbook.

Task 2.3. CNOT_{12} or SWAP?

Input: An operation that implements a two-qubit unitary transformation: either the CNOT gate with the first qubit as control and the second qubit as target (CNOT_{12}) or the $SWAP$ gate.

- The operation will accept an array of qubits as input, but it will fail if the array is empty or has one or more than two qubits.
- The operation will have Adjoint and Controlled variants defined.

Output: 0 if the given operation is CNOT_{12}, 1 if the given operation is SWAP.

You are allowed to apply the given operation and its adjoint/controlled variants exactly **once**.

In [ ]:
```
%kata T203_DistinguishCNOTfromSWAP

operation DistinguishCNOTfromSWAP (unitary : (Qubit[] => Unit is Adj+Ctl)) : I
    // ...
    return -1;
}
```

Can't come up with a solution? See the explained solution in the Distinguish Unitaries Workbook.

Task 2.4. Identity, CNOTs or SWAP?

Input: An operation that implements a two-qubit unitary transformation: either the identity ($I \otimes I$), the CNOT gate with one of the qubits as control and the other qubit as a target, or the $SWAP$ gate.

- The operation will accept an array of qubits as input, but it will fail if the array is empty or has one or more than two qubits.
- The operation will have Adjoint and Controlled variants defined.

Output:

- 0 if the given operation is $I \otimes I$,
- 1 if the given operation is CNOT_{12},
- 2 if the given operation is CNOT_{21},
- 3 if the given operation is SWAP.

You are allowed to apply the given operation and its adjoint/controlled variants at most **twice**.

In [ ]:
```
%kata T204_DistinguishTwoQubitUnitaries

operation DistinguishTwoQubitUnitaries (unitary : (Qubit[] => Unit is Adj+Ctl)
    // ...
    return -1;
}
```

Can't come up with a solution? See the explained solution in the Distinguish Unitaries Workbook.