Practical 9

Name: Heja Bibani
Student Number: 16301173

In [2]:

```
from qiskit import *
from qiskit.providers.aer import QasmSimulator
from qiskit.visualization import plot_histogram
import numpy as np
from math import sqrt
%matplotlib inline
```

Section 1

Let matrix $A = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix}, B = \begin{bmatrix} \frac{1}{2} & -\frac{\sqrt{3}}{2} \\ \frac{\sqrt{3}}{2} & \frac{1}{2} \end{bmatrix}$. Use these two matrices to show that the Kronecker Product of matrices may not satisfy the commutativity: $A \otimes B = B \otimes A$.

$$A \otimes B = \begin{bmatrix} \frac{1}{\sqrt{2}} \times B & \frac{1}{\sqrt{2}} \times B \\ \frac{1}{\sqrt{2}} \times B & -\frac{1}{\sqrt{2}} \times B \end{bmatrix} = \begin{bmatrix} \frac{1}{2\sqrt{2}} & -\frac{\sqrt{3}}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & -\frac{\sqrt{3}}{2\sqrt{2}} \\ \frac{\sqrt{3}}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{\sqrt{3}}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} \\ \frac{1}{2\sqrt{2}} & -\frac{\sqrt{3}}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & \frac{\sqrt{3}}{2\sqrt{2}} \\ \frac{\sqrt{3}}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & -\frac{\sqrt{3}}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} \end{bmatrix}$$

$$B \otimes A = \begin{bmatrix} \frac{1}{2} \times A & -\frac{\sqrt{3}}{2} \times A \\ \frac{\sqrt{3}}{2} \times A & \frac{1}{2} \times A \end{bmatrix} = \begin{bmatrix} \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & -\frac{\sqrt{3}}{2\sqrt{2}} & -\frac{\sqrt{3}}{2\sqrt{2}} \\ \frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & -\frac{\sqrt{3}}{2\sqrt{2}} & \frac{\sqrt{3}}{2\sqrt{2}} \\ \frac{\sqrt{3}}{2\sqrt{2}} & \frac{\sqrt{3}}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} \\ \frac{\sqrt{3}}{2\sqrt{2}} & -\frac{\sqrt{3}}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} \end{bmatrix}$$

We note here that $A \otimes B \neq B \otimes A$, and does not satisfy commutativity. Below we will test this by utilizing python to confirm the manual calculation.

In [3]:

```
a = np.array([[1/sqrt(2),1/sqrt(2)],[1/sqrt(2),-1/sqrt(2)]])
a
b = np.array([[1/2,-sqrt(3)/2],[sqrt(3)/2,1/2]])
b
```

Out[3]:

```
array([[ 0.5       , -0.8660254],
       [ 0.8660254,  0.5       ]])
```

In [4]:

```
prod_ab = np.kron(a, b)
prod_ab
```

Out[4]:

```
array([[ 0.35355339, -0.61237244,  0.35355339, -0.61237244],
       [ 0.61237244,  0.35355339,  0.61237244,  0.35355339],
       [ 0.35355339, -0.61237244, -0.35355339,  0.61237244],
       [ 0.61237244,  0.35355339, -0.61237244, -0.35355339]])
```

In [5]:

```
prod_ba = np.kron(b, a)
prod_ba
```

Out[5]:

```
array([[ 0.35355339,  0.35355339, -0.61237244, -0.61237244],
       [ 0.35355339, -0.35355339, -0.61237244,  0.61237244],
       [ 0.61237244,  0.61237244,  0.35355339,  0.35355339],
       [ 0.61237244, -0.61237244,  0.35355339, -0.35355339]])
```

We can see above that the arrays are indeed not equal and thus confirms the results completed in the manual calculation and $A \otimes B \neq B \otimes A$.

Section 2

Let matrix $A = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix}, B = \begin{bmatrix} \frac{1}{2} & -\frac{\sqrt{3}}{2} \\ \frac{\sqrt{3}}{2} & \frac{1}{2} \end{bmatrix}, C = \begin{bmatrix} -\frac{1}{2} & -\frac{\sqrt{3}}{2} \\ \frac{\sqrt{3}}{2} & -\frac{1}{2} \end{bmatrix}$. Use these three matrices to show that the Kronecker Product of matrices satisfies the Associativity: $(A \otimes B) \otimes C = A \otimes (B \otimes C)$.

$(A \otimes B) \otimes C$ =

$$A \otimes B = \begin{bmatrix} \frac{1}{\sqrt{2}} \times B & \frac{1}{\sqrt{2}} \times B \\ \frac{1}{\sqrt{2}} \times B & -\frac{1}{\sqrt{2}} \times B \end{bmatrix} = \begin{bmatrix} \frac{1}{2\sqrt{2}} & -\frac{\sqrt{3}}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & -\frac{\sqrt{3}}{2\sqrt{2}} \\ \frac{\sqrt{3}}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{\sqrt{3}}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} \\ \frac{1}{2\sqrt{2}} & -\frac{\sqrt{3}}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & \frac{\sqrt{3}}{2\sqrt{2}} \\ \frac{\sqrt{3}}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & -\frac{\sqrt{3}}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} \end{bmatrix}$$

$$(A \otimes B) \otimes C = \begin{bmatrix} \frac{1}{2\sqrt{2}} \times C & -\frac{\sqrt{3}}{2\sqrt{2}} \times C & \frac{1}{2\sqrt{2}} \times C & -\frac{\sqrt{3}}{2\sqrt{2}} \times C \\ \frac{\sqrt{3}}{2\sqrt{2}} \times C & \frac{1}{2\sqrt{2}} \times C & \frac{\sqrt{3}}{2\sqrt{2}} \times C & \frac{1}{2\sqrt{2}} \times C \\ \frac{1}{2\sqrt{2}} \times C & -\frac{\sqrt{3}}{2\sqrt{2}} \times C & -\frac{1}{2\sqrt{2}} \times C & \frac{\sqrt{3}}{2\sqrt{2}} \times C \\ \frac{\sqrt{3}}{2\sqrt{2}} \times C & \frac{1}{2\sqrt{2}} \times C & -\frac{\sqrt{3}}{2\sqrt{2}} \times C & -\frac{1}{2\sqrt{2}} \times C \end{bmatrix}$$

$$= \begin{bmatrix} -\frac{1}{4\sqrt{2}} & -\frac{\sqrt{3}}{4\sqrt{2}} & \frac{\sqrt{3}}{4\sqrt{2}} & \frac{3}{4\sqrt{2}} & -\frac{1}{4\sqrt{2}} & -\frac{\sqrt{3}}{4\sqrt{2}} & \frac{\sqrt{3}}{4\sqrt{2}} & \frac{3}{4\sqrt{2}} \\ \frac{\sqrt{3}}{4\sqrt{2}} & -\frac{1}{4\sqrt{2}} & -\frac{3}{4\sqrt{2}} & \frac{\sqrt{3}}{4\sqrt{2}} & \frac{\sqrt{3}}{4\sqrt{2}} & -\frac{1}{4\sqrt{2}} & -\frac{3}{4\sqrt{2}} & \frac{\sqrt{3}}{4\sqrt{2}} \\ -\frac{\sqrt{3}}{4\sqrt{2}} & -\frac{3}{4\sqrt{2}} & -\frac{1}{4\sqrt{2}} & -\frac{\sqrt{3}}{4\sqrt{2}} & -\frac{\sqrt{3}}{4\sqrt{2}} & -\frac{3}{4\sqrt{2}} & -\frac{1}{4\sqrt{2}} & -\frac{\sqrt{3}}{4\sqrt{2}} \\ \frac{3}{4\sqrt{2}} & -\frac{\sqrt{3}}{4\sqrt{2}} & \frac{\sqrt{3}}{4\sqrt{2}} & -\frac{1}{4\sqrt{2}} & \frac{3}{4\sqrt{2}} & -\frac{\sqrt{3}}{4\sqrt{2}} & \frac{\sqrt{3}}{4\sqrt{2}} & -\frac{1}{4\sqrt{2}} \\ -\frac{1}{4\sqrt{2}} & -\frac{\sqrt{3}}{4\sqrt{2}} & \frac{\sqrt{3}}{4\sqrt{2}} & \frac{3}{4\sqrt{2}} & \frac{1}{4\sqrt{2}} & \frac{\sqrt{3}}{4\sqrt{2}} & -\frac{\sqrt{3}}{4\sqrt{2}} & -\frac{3}{4\sqrt{2}} \\ \frac{\sqrt{3}}{4\sqrt{2}} & -\frac{1}{4\sqrt{2}} & -\frac{3}{4\sqrt{2}} & \frac{\sqrt{3}}{4\sqrt{2}} & -\frac{\sqrt{3}}{4\sqrt{2}} & \frac{1}{4\sqrt{2}} & \frac{3}{4\sqrt{2}} & -\frac{\sqrt{3}}{4\sqrt{2}} \\ -\frac{\sqrt{3}}{4\sqrt{2}} & -\frac{3}{4\sqrt{2}} & -\frac{1}{4\sqrt{2}} & -\frac{\sqrt{3}}{4\sqrt{2}} & \frac{\sqrt{3}}{4\sqrt{2}} & \frac{3}{4\sqrt{2}} & \frac{1}{4\sqrt{2}} & \frac{\sqrt{3}}{4\sqrt{2}} \\ \frac{3}{4\sqrt{2}} & -\frac{\sqrt{3}}{4\sqrt{2}} & \frac{\sqrt{3}}{4\sqrt{2}} & -\frac{1}{4\sqrt{2}} & -\frac{3}{4\sqrt{2}} & \frac{\sqrt{3}}{4\sqrt{2}} & -\frac{\sqrt{3}}{4\sqrt{2}} & \frac{1}{4\sqrt{2}} \end{bmatrix}$$

$A \otimes (B \otimes C)$ =

$$B \otimes C = \begin{bmatrix} \frac{1}{2} \times C & -\frac{\sqrt{3}}{2} \times C \\ \frac{\sqrt{3}}{2} \times C & \frac{1}{2} \times C \end{bmatrix} = \begin{bmatrix} -\frac{1}{4} & -\frac{\sqrt{3}}{4} & \frac{\sqrt{3}}{4} & \frac{3}{4} \\ \frac{\sqrt{3}}{4} & -\frac{1}{4} & -\frac{3}{4} & \frac{\sqrt{3}}{4} \\ -\frac{\sqrt{3}}{4} & -\frac{3}{4} & -\frac{1}{4} & -\frac{\sqrt{3}}{4} \\ \frac{3}{4} & -\frac{\sqrt{3}}{4} & \frac{\sqrt{3}}{4} & -\frac{1}{4} \end{bmatrix}$$

$$A \otimes (B \otimes C) = \begin{bmatrix} \frac{1}{\sqrt{2}} \times (B \otimes C) & \frac{1}{\sqrt{2}} \times (B \otimes C) \\ \frac{1}{\sqrt{2}} \times (B \otimes C) & -\frac{1}{\sqrt{2}} \times (B \otimes C) \end{bmatrix}$$

$$= \begin{bmatrix} -\frac{1}{4\sqrt{2}} & -\frac{\sqrt{3}}{4\sqrt{2}} & \frac{\sqrt{3}}{4\sqrt{2}} & \frac{3}{4\sqrt{2}} & -\frac{1}{4\sqrt{2}} & -\frac{\sqrt{3}}{4\sqrt{2}} & \frac{\sqrt{3}}{4\sqrt{2}} & \frac{3}{4\sqrt{2}} \\ \frac{\sqrt{3}}{4\sqrt{2}} & -\frac{1}{4\sqrt{2}} & -\frac{3}{4\sqrt{2}} & \frac{\sqrt{3}}{4\sqrt{2}} & \frac{\sqrt{3}}{4\sqrt{2}} & -\frac{1}{4\sqrt{2}} & -\frac{3}{4\sqrt{2}} & \frac{\sqrt{3}}{4\sqrt{2}} \\ -\frac{\sqrt{3}}{4\sqrt{2}} & -\frac{3}{4\sqrt{2}} & -\frac{1}{4\sqrt{2}} & -\frac{\sqrt{3}}{4\sqrt{2}} & -\frac{\sqrt{3}}{4\sqrt{2}} & -\frac{3}{4\sqrt{2}} & -\frac{1}{4\sqrt{2}} & -\frac{\sqrt{3}}{4\sqrt{2}} \\ \frac{3}{4\sqrt{2}} & -\frac{\sqrt{3}}{4\sqrt{2}} & \frac{\sqrt{3}}{4\sqrt{2}} & -\frac{1}{4\sqrt{2}} & \frac{3}{4\sqrt{2}} & -\frac{\sqrt{3}}{4\sqrt{2}} & \frac{\sqrt{3}}{4\sqrt{2}} & -\frac{1}{4\sqrt{2}} \\ -\frac{1}{4\sqrt{2}} & -\frac{\sqrt{3}}{4\sqrt{2}} & \frac{\sqrt{3}}{4\sqrt{2}} & \frac{3}{4\sqrt{2}} & \frac{1}{4\sqrt{2}} & \frac{\sqrt{3}}{4\sqrt{2}} & -\frac{\sqrt{3}}{4\sqrt{2}} & -\frac{3}{4\sqrt{2}} \\ \frac{\sqrt{3}}{4\sqrt{2}} & -\frac{1}{4\sqrt{2}} & -\frac{3}{4\sqrt{2}} & \frac{\sqrt{3}}{4\sqrt{2}} & -\frac{\sqrt{3}}{4\sqrt{2}} & \frac{1}{4\sqrt{2}} & \frac{3}{4\sqrt{2}} & -\frac{\sqrt{3}}{4\sqrt{2}} \\ -\frac{\sqrt{3}}{4\sqrt{2}} & -\frac{3}{4\sqrt{2}} & -\frac{1}{4\sqrt{2}} & -\frac{\sqrt{3}}{4\sqrt{2}} & \frac{\sqrt{3}}{4\sqrt{2}} & \frac{3}{4\sqrt{2}} & \frac{1}{4\sqrt{2}} & \frac{\sqrt{3}}{4\sqrt{2}} \\ \frac{3}{4\sqrt{2}} & -\frac{\sqrt{3}}{4\sqrt{2}} & \frac{\sqrt{3}}{4\sqrt{2}} & -\frac{1}{4\sqrt{2}} & -\frac{3}{4\sqrt{2}} & \frac{\sqrt{3}}{4\sqrt{2}} & -\frac{\sqrt{3}}{4\sqrt{2}} & \frac{1}{4\sqrt{2}} \end{bmatrix}$$

The two matricies are equal and thus $(A \otimes B) \otimes C = A \otimes (B \otimes C)$ and satisifies associativity. Below we will check the results via python to confirm the calculation is correct.

In [6]:

```
a = np.array([[1/sqrt(2),1/sqrt(2)],[1/sqrt(2),-1/sqrt(2)]])
a
b = np.array([[1/2,-sqrt(3)/2],[sqrt(3)/2,1/2]])
b
c = np.array([[-1/2,-sqrt(3)/2],[sqrt(3)/2,-1/2]])
c
```

Out[6]:

```
array([[-0.5       , -0.8660254],
       [ 0.8660254, -0.5       ]])
```

In [7]:

```
prod_ab = np.kron(a, b)
prod_ab_c = np.kron(prod_ab, c)
prod_ab_c
```

Out[7]:

```
array([[-0.1767767 , -0.30618622,  0.30618622,  0.53033009, -0.1767767 ,
        -0.30618622,  0.30618622,  0.53033009],
       [ 0.30618622, -0.1767767 , -0.53033009,  0.30618622,  0.30618622,
        -0.1767767 , -0.53033009,  0.30618622],
       [-0.30618622, -0.53033009, -0.1767767 , -0.30618622, -0.30618622,
        -0.53033009, -0.1767767 , -0.30618622],
       [ 0.53033009, -0.30618622,  0.30618622, -0.1767767 ,  0.53033009,
        -0.30618622,  0.30618622, -0.1767767 ],
       [-0.1767767 , -0.30618622,  0.30618622,  0.53033009,  0.1767767 ,
         0.30618622, -0.30618622, -0.53033009],
       [ 0.30618622, -0.1767767 , -0.53033009,  0.30618622, -0.30618622,
         0.1767767 ,  0.53033009, -0.30618622],
       [-0.30618622, -0.53033009, -0.1767767 , -0.30618622,  0.30618622,
         0.53033009,  0.1767767 ,  0.30618622],
       [ 0.53033009, -0.30618622,  0.30618622, -0.1767767 , -0.53033009,
         0.30618622, -0.30618622,  0.1767767 ]])
```

In [8]:

```
prod_bc = np.kron(b,c)
prod_a_bc = np.kron(a, prod_bc)
prod_a_bc
```

Out[8]:

```
array([[-0.1767767 , -0.30618622,  0.30618622,  0.53033009, -0.1767767 ,
        -0.30618622,  0.30618622,  0.53033009],
       [ 0.30618622, -0.1767767 , -0.53033009,  0.30618622,  0.30618622,
        -0.1767767 , -0.53033009,  0.30618622],
       [-0.30618622, -0.53033009, -0.1767767 , -0.30618622, -0.30618622,
        -0.53033009, -0.1767767 , -0.30618622],
       [ 0.53033009, -0.30618622,  0.30618622, -0.1767767 ,  0.53033009,
        -0.30618622,  0.30618622, -0.1767767 ],
       [-0.1767767 , -0.30618622,  0.30618622,  0.53033009,  0.1767767 ,
         0.30618622, -0.30618622, -0.53033009],
       [ 0.30618622, -0.1767767 , -0.53033009,  0.30618622, -0.30618622,
         0.1767767 ,  0.53033009, -0.30618622],
       [-0.30618622, -0.53033009, -0.1767767 , -0.30618622,  0.30618622,
         0.53033009,  0.1767767 ,  0.30618622],
       [ 0.53033009, -0.30618622,  0.30618622, -0.1767767 , -0.53033009,
         0.30618622, -0.30618622,  0.1767767 ]])
```

We can see above that this matches the expected results in the manual calculation and satisifies Associativity. Thus $(A \otimes B) \otimes C = A \otimes (B \otimes C)$.

Section 3

3.1

Derive the input/output table for its oracle F_0. (Search the Internet to learn how to do a table with Markdown)

$/y\rangle, /x\rangle, /z\rangle$	$/y \oplus F_0(x)\rangle, /x\rangle, /z\rangle$		**Output**
$\lvert 0\rangle, \lvert 0\rangle, \lvert 0\rangle$	$\lvert 0 \oplus F_0(0,0)\rangle, \lvert 0\rangle$	$\lvert 0 \oplus 0\rangle, \lvert 0\rangle, \lvert 0\rangle$	$\lvert 0\rangle, \lvert 0\rangle, \lvert 0\rangle$
$\lvert 0\rangle, \lvert 0\rangle, \lvert 1\rangle$	$\lvert 0 \oplus F_0(0,1)\rangle, \lvert 1\rangle$	$\lvert 0 \oplus 0\rangle, \lvert 0\rangle, \lvert 1\rangle$	$\lvert 0\rangle, \lvert 0\rangle, \lvert 1\rangle$
$\lvert 0\rangle, \lvert 1\rangle, \lvert 0\rangle$	$\lvert 1 \oplus F_0(1,0)\rangle, \lvert 0\rangle$	$\lvert 0 \oplus 0\rangle, \lvert 1\rangle, \lvert 0\rangle$	$\lvert 0\rangle, \lvert 1\rangle, \lvert 0\rangle$
$\lvert 0\rangle, \lvert 1\rangle, \lvert 1\rangle$	$\lvert 1 \oplus F_0(1,1)\rangle, \lvert 1\rangle$	$\lvert 0 \oplus 0\rangle, \lvert 1\rangle, \lvert 1\rangle$	$\lvert 0\rangle, \lvert 1\rangle, \lvert 1\rangle$
$\lvert 1\rangle, \lvert 0\rangle, \lvert 0\rangle$	$\lvert 0 \oplus F_0(0,0)\rangle, \lvert 0\rangle$	$\lvert 1 \oplus 0\rangle, \lvert 0\rangle, \lvert 0\rangle$	$\lvert 1\rangle, \lvert 0\rangle, \lvert 0\rangle$
$\lvert 1\rangle, \lvert 0\rangle, \lvert 1\rangle$	$\lvert 0 \oplus F_0(0,1)\rangle, \lvert 1\rangle$	$\lvert 1 \oplus 0\rangle, \lvert 0\rangle, \lvert 1\rangle$	$\lvert 1\rangle, \lvert 0\rangle, \lvert 1\rangle$
$\lvert 1\rangle, \lvert 1\rangle, \lvert 0\rangle$	$\lvert 1 \oplus F_0(1,0)\rangle, \lvert 0\rangle$	$\lvert 1 \oplus 0\rangle, \lvert 1\rangle, \lvert 0\rangle$	$\lvert 1\rangle, \lvert 1\rangle, \lvert 0\rangle$
$\lvert 1\rangle, \lvert 1\rangle, \lvert 1\rangle$	$\lvert 1 \oplus F_0(1,1)\rangle, \lvert 1\rangle$	$\lvert 1 \oplus 0\rangle, \lvert 1\rangle, \lvert 1\rangle$	$\lvert 1\rangle, \lvert 1\rangle, \lvert 1\rangle$

3.2

Derive the gate matrix for F_0

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

3.3

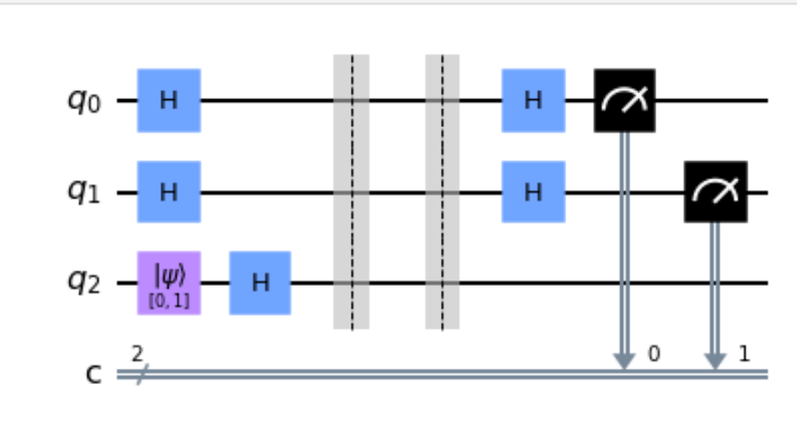
Use a Quantum Circuit to implement the oracle F_0, and draw this circuit.

Since the gate is the identity then we don't implement anything for this particular circuit.

In [9]:

```
# Construct a circuit with a 3-qubit input, and 2-bit measurement storage.
qc = QuantumCircuit(3, 2)

# Initialise q2 with |1>
initial_state = [0, 1]
qc.initialize(initial_state, 2)

# Apply H-gates to q0, q1 and q2
qc.h(0)
qc.h(1)
qc.h(2)

# Insert the oracle F1
# And add two barriers to enclose this oracle.
qc.barrier()
qc.barrier()

# Apply H-gates to q0, q1
qc.h(0)
qc.h(1)

# Measure q0, q1
qc.measure([0,1], [0,1])

# Draw the circuit
qc.draw()
```

Out[9]:



Section 4

4.1

Derive the input/output table for its oracle F_4.

$/y\rangle, /x\rangle, /z\rangle$	$/y \oplus F_4(x)\rangle, /x\rangle, /z\rangle$		**Output**
$\lvert 0\rangle, \lvert 0\rangle, \lvert 0\rangle$	$\lvert 0 \oplus F_4(0,0)\rangle, \lvert 0\rangle, \lvert 0\rangle$	$\lvert 0 \oplus 0\rangle, \lvert 0\rangle, \lvert 0\rangle$	$\lvert 0\rangle, \lvert 0\rangle, \lvert 0\rangle$
$\lvert 0\rangle, \lvert 0\rangle, \lvert 1\rangle$	$\lvert 0 \oplus F_4(0,1)\rangle, \lvert 0\rangle, \lvert 1\rangle$	$\lvert 0 \oplus 1\rangle, \lvert 0\rangle, \lvert 1\rangle$	$\lvert 1\rangle, \lvert 0\rangle, \lvert 1\rangle$
$\lvert 0\rangle, \lvert 1\rangle, \lvert 0\rangle$	$\lvert 1 \oplus F_4(1,0)\rangle, \lvert 1\rangle, \lvert 0\rangle$	$\lvert 0 \oplus 0\rangle, \lvert 1\rangle, \lvert 0\rangle$	$\lvert 0\rangle, \lvert 1\rangle, \lvert 0\rangle$
$\lvert 0\rangle, \lvert 1\rangle, \lvert 1\rangle$	$\lvert 1 \oplus F_4(1,1)\rangle, \lvert 1\rangle, \lvert 1\rangle$	$\lvert 0 \oplus 0\rangle, \lvert 1\rangle, \lvert 1\rangle$	$\lvert 0\rangle, \lvert 1\rangle, \lvert 1\rangle$
$\lvert 1\rangle, \lvert 0\rangle, \lvert 0\rangle$	$\lvert 0 \oplus F_4(0,0)\rangle, \lvert 0\rangle, \lvert 0\rangle$	$\lvert 1 \oplus 0\rangle, \lvert 0\rangle, \lvert 0\rangle$	$\lvert 1\rangle, \lvert 0\rangle, \lvert 0\rangle$
$\lvert 1\rangle, \lvert 0\rangle, \lvert 1\rangle$	$\lvert 0 \oplus F_4(0,1)\rangle, \lvert 0\rangle, \lvert 1\rangle$	$\lvert 1 \oplus 1\rangle, \lvert 0\rangle, \lvert 1\rangle$	$\lvert 0\rangle, \lvert 0\rangle, \lvert 1\rangle$
$\lvert 1\rangle, \lvert 1\rangle, \lvert 0\rangle$	$\lvert 1 \oplus F_4(1,0)\rangle, \lvert 1\rangle, \lvert 0\rangle$	$\lvert 1 \oplus 0\rangle, \lvert 1\rangle, \lvert 0\rangle$	$\lvert 1\rangle, \lvert 1\rangle, \lvert 0\rangle$
$\lvert 1\rangle, \lvert 1\rangle, \lvert 1\rangle$	$\lvert 1 \oplus F_4(1,1)\rangle, \lvert 1\rangle, \lvert 1\rangle$	$\lvert 1 \oplus 0\rangle, \lvert 1\rangle, \lvert 1\rangle$	$\lvert 1\rangle, \lvert 1\rangle, \lvert 1\rangle$

4.2

Derive the gate matrix for F_4

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

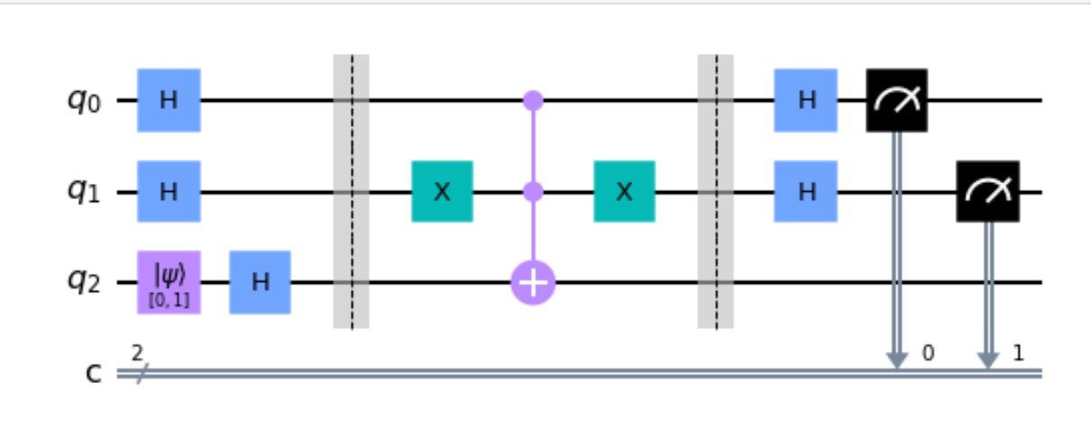
4.3

This gate is a variation of the toffoli gate where we place a NOT gate on the x_1 value and then another to re-instate its value.

In [10]:

```
# Construct a circuit with a 3-qubit input, and 2-bit measurement storage.
qc = QuantumCircuit(3, 2)

# Initialise q2 with |1>
initial_state = [0, 1]
qc.initialize(initial_state, 2)

# Apply H-gates to q0, q1 and q2
qc.h(0)
qc.h(1)
qc.h(2)

# Insert the oracle F1
# And add two barriers to enclose this oracle.
qc.barrier()
qc.x(1)
qc.ccx(0,1,2)
qc.x(1)
qc.barrier()

# Apply H-gates to q0, q1
qc.h(0)
qc.h(1)

# Measure q0, q1
qc.measure([0,1], [0,1])

# Draw the circuit
qc.draw()
```

Out[10]:



Section 5

5.1

Derive the input/output table for its oracle F_5.

$/y\rangle, /x\rangle, /z\rangle$	$/y \oplus F_5(x)\rangle, /x\rangle, /z\rangle$		**Output**
$\lvert 0\rangle, \lvert 0\rangle, \lvert 0\rangle$	$\lvert 0 \oplus F_5(0,0)\rangle, \lvert 0\rangle, \lvert 0\rangle$	$\lvert 0 \oplus 0\rangle, \lvert 0\rangle, \lvert 0\rangle$	$\lvert 0\rangle, \lvert 0\rangle, \lvert 0\rangle$
$\lvert 0\rangle, \lvert 0\rangle, \lvert 1\rangle$	$\lvert 0 \oplus F_5(0,1)\rangle, \lvert 0\rangle, \lvert 1\rangle$	$\lvert 0 \oplus 1\rangle, \lvert 0\rangle, \lvert 1\rangle$	$\lvert 1\rangle, \lvert 0\rangle, \lvert 1\rangle$
$\lvert 0\rangle, \lvert 1\rangle, \lvert 0\rangle$	$\lvert 1 \oplus F_5(1,0)\rangle, \lvert 1\rangle, \lvert 0\rangle$	$\lvert 0 \oplus 0\rangle, \lvert 1\rangle, \lvert 0\rangle$	$\lvert 0\rangle, \lvert 1\rangle, \lvert 0\rangle$
$\lvert 0\rangle, \lvert 1\rangle, \lvert 1\rangle$	$\lvert 1 \oplus F_5(1,1)\rangle, \lvert 1\rangle, \lvert 1\rangle$	$\lvert 0 \oplus 1\rangle, \lvert 1\rangle, \lvert 1\rangle$	$\lvert 1\rangle, \lvert 1\rangle, \lvert 1\rangle$
$\lvert 1\rangle, \lvert 0\rangle, \lvert 0\rangle$	$\lvert 0 \oplus F_5(0,0)\rangle, \lvert 0\rangle, \lvert 0\rangle$	$\lvert 1 \oplus 0\rangle, \lvert 0\rangle, \lvert 0\rangle$	$\lvert 1\rangle, \lvert 0\rangle, \lvert 0\rangle$
$\lvert 1\rangle, \lvert 0\rangle, \lvert 1\rangle$	$\lvert 0 \oplus F_5(0,1)\rangle, \lvert 0\rangle, \lvert 1\rangle$	$\lvert 1 \oplus 1\rangle, \lvert 0\rangle, \lvert 1\rangle$	$\lvert 0\rangle, \lvert 0\rangle, \lvert 1\rangle$
$\lvert 1\rangle, \lvert 1\rangle, \lvert 0\rangle$	$\lvert 1 \oplus F_5(1,0)\rangle, \lvert 1\rangle, \lvert 0\rangle$	$\lvert 1 \oplus 0\rangle, \lvert 1\rangle, \lvert 0\rangle$	$\lvert 1\rangle, \lvert 1\rangle, \lvert 0\rangle$
$\lvert 1\rangle, \lvert 1\rangle, \lvert 1\rangle$	$\lvert 1 \oplus F_5(1,1)\rangle, \lvert 1\rangle, \lvert 1\rangle$	$\lvert 1 \oplus 1\rangle, \lvert 1\rangle, \lvert 1\rangle$	$\lvert 0\rangle, \lvert 1\rangle, \lvert 1\rangle$

5.2

Derive the gate matrix for F_5.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

5.3

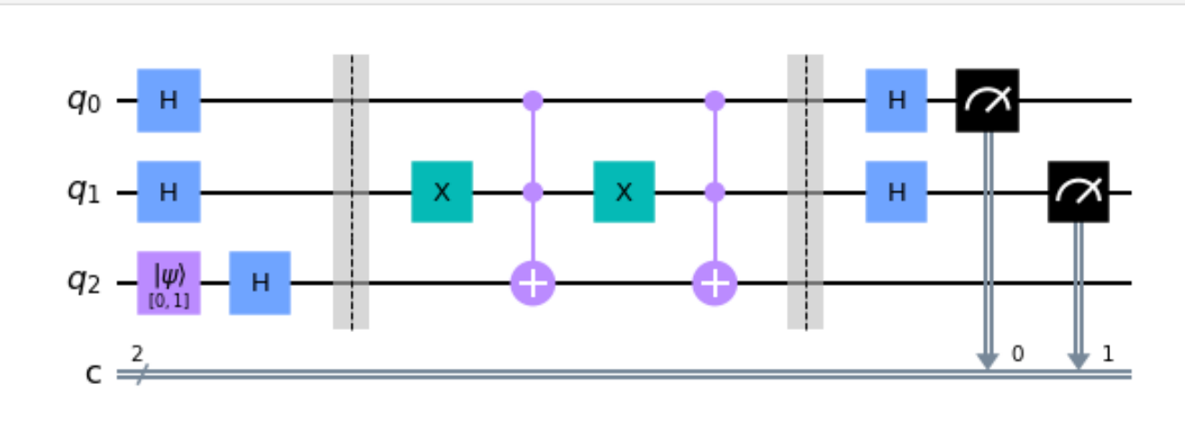
Use a Quantum Circuit to implement the oracle F_5, and draw this circuit.

This particular gate first uses the implementation of F_4 to swap the positions of bits '001' and '101'. When that is done, the positions '011' and '111' needs to be swapped. Adding a second taffoli gate will not swap the position '001' and '101' again, then the only positions that will be swapped after applying the second taffoli gate will switch the remaining positions of '011' and '111'. Thus satisfying the requirements described in 5.1 and 5.2.

In [11]:

```
# Construct a circuit with a 3-qubit input, and 2-bit measurement storage.
qc = QuantumCircuit(3, 2)

# Initialise q2 with |1>
initial_state = [0, 1]
qc.initialize(initial_state, 2)

# Apply H-gates to q0, q1 and q2
qc.h(0)
qc.h(1)
qc.h(2)

# Insert the oracle F1
# And add two barriers to enclose this oracle.
qc.barrier()
qc.x(1)
qc.ccx(0,1,2)
qc.x(1)
qc.ccx(0,1,2)
qc.barrier()

# Apply H-gates to q0, q1
qc.h(0)
qc.h(1)

# Measure q0, q1
qc.measure([0,1], [0,1])

# Draw the circuit
qc.draw()
```

Out[11]:



Section 6

Use the QasmSimulator to implement the Deutsch-Jozsa Algorithm with the oracles F_0, F_4, and F_5 above respectively. For each oracle, the implementation should roughly follow the structure of the notebook accompanying our lecture, consisting of measuring the outcomes with 1000 shots and plotting the results with histograms.

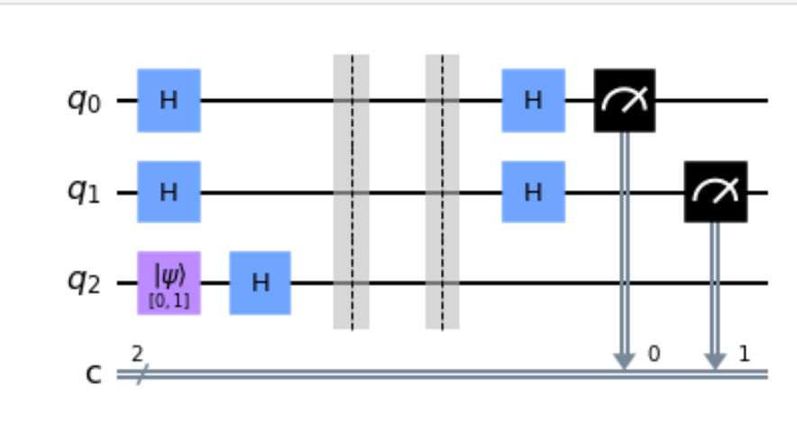
Task 1: For F_0

This is a constant oracle so we would expect that the value would be '00' with 100% probability.

In [12]:

```
# Construct a circuit with a 3-qubit input, and 2-bit measurement storage.
qc = QuantumCircuit(3, 2)

# Initialise q2 with |1>
initial_state = [0, 1]
qc.initialize(initial_state, 2)

# Apply H-gates to q0, q1 and q2
qc.h(0)
qc.h(1)
qc.h(2)

# Insert the oracle F1
# And add two barriers to enclose this oracle.
qc.barrier()
qc.barrier()

# Apply H-gates to q0, q1
qc.h(0)
qc.h(1)

# Measure q0, q1
qc.measure([0,1], [0,1])

# Draw the circuit
qc.draw()
```

Out[12]:

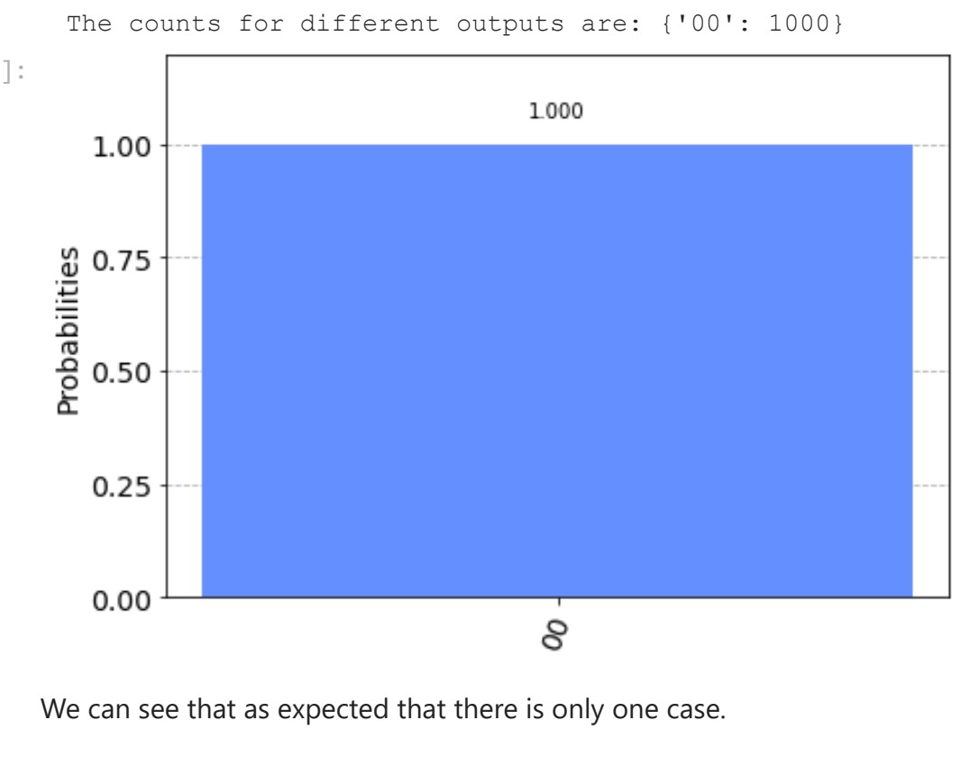


In [13]:

```
simulator = QasmSimulator()
# Run the circuit with certain number of shots.
# The 'shots' below means the number of times to run this circuit.
# Its default value is 1024, so if it is omitted, you'll run the circuit 1024 times.
job = execute(qc, backend=simulator, shots=1000)

# Call result() to obtain the result
result = job.result()

# Call get_counts() to obtain the counts of different outputs
counts = result.get_counts(qc)
# NB: 'counts' has a type of <class 'qiskit.result.counts.Counts'>,
# which is similar to a type of dictionary.
print(type(counts))

# Print the counts of different outputs
print("\n The counts for different outputs are:", counts)

# Plot the histogram
plot_histogram(counts)
```

```
<class 'qiskit.result.counts.Counts'>

 The counts for different outputs are: {'00': 1000}
```

Out[13]:



We can see that as expected that there is only one case.

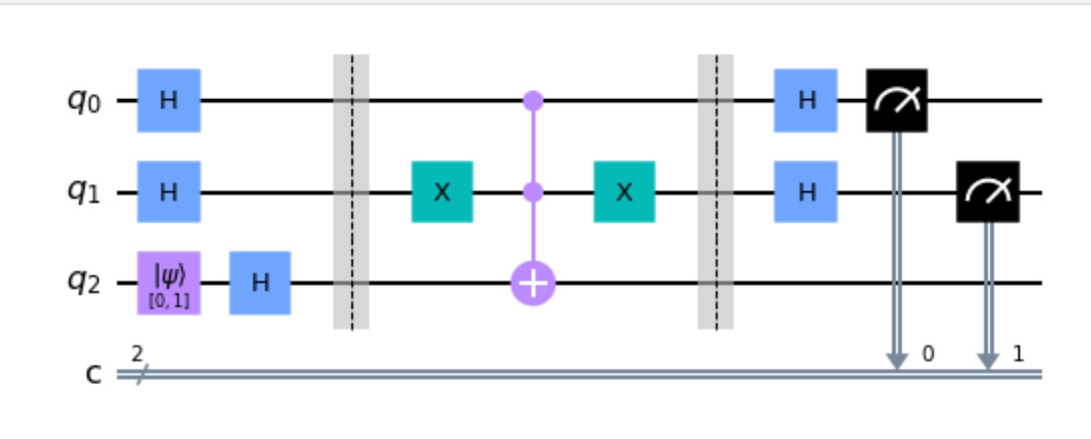
Task 2: For F_4

This is neither a balanced or constant oracle thus we would expect that '00' won't appear with 100% certainity. We'll demonstrate that, in this case, the DJ algorithm cannot conclude anything.

In [14]:

```
# Construct a circuit with a 3-qubit input, and 2-bit measurement storage.
qc = QuantumCircuit(3, 2)

# Initialise q2 with |1>
initial_state = [0, 1]
qc.initialize(initial_state, 2)

# Apply H-gates to q0, q1 and q2
qc.h(0)
qc.h(1)
qc.h(2)

# Insert the oracle F1
# And add two barriers to enclose this oracle.
qc.barrier()
qc.x(1)
qc.ccx(0,1,2)
qc.x(1)
qc.barrier()

# Apply H-gates to q0, q1
qc.h(0)
qc.h(1)

# Measure q0, q1
qc.measure([0,1], [0,1])

# Draw the circuit
qc.draw()
```

Out[14]:

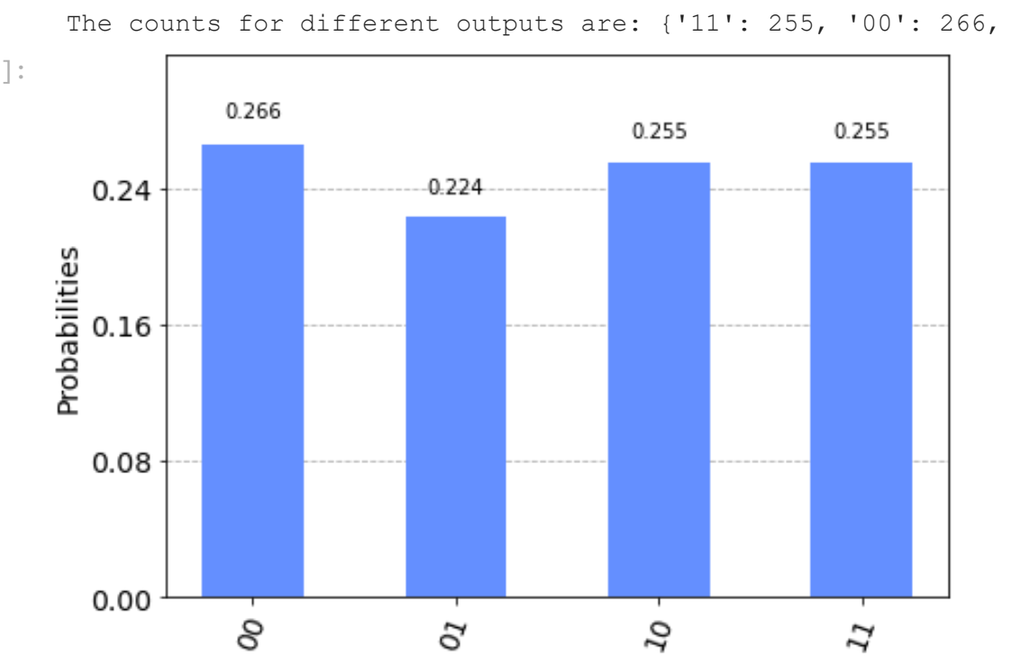


In [15]:

```
simulator = QasmSimulator()
# Run the circuit with certain number of shots.
# The 'shots' below means the number of times to run this circuit.
# Its default value is 1024, so if it is omitted, you'll run the circuit 1024 times.
job = execute(qc, backend=simulator, shots=1000)

# Call result() to obtain the result
result = job.result()

# Call get_counts() to obtain the counts of different outputs
counts = result.get_counts(qc)
# NB: 'counts' has a type of <class 'qiskit.result.counts.Counts'>,
# which is similar to a type of dictionary.
print(type(counts))

# Print the counts of different outputs
print("\n The counts for different outputs are:", counts)

# Plot the histogram
plot_histogram(counts)
```

```
<class 'qiskit.result.counts.Counts'>

 The counts for different outputs are: {'11': 255, '00': 266, '10': 255, '01': 224}
```

Out[15]:



So we see that '00' indeed appears with a probability of about 0.25. Thus, '00' is sometimes the measurement outcome, but sometimes not. Therefore, the DJ algorithm cannot conclude anything based on whether the measurement outcome is '00'. With calculation we confirmed that this was indeed the result.

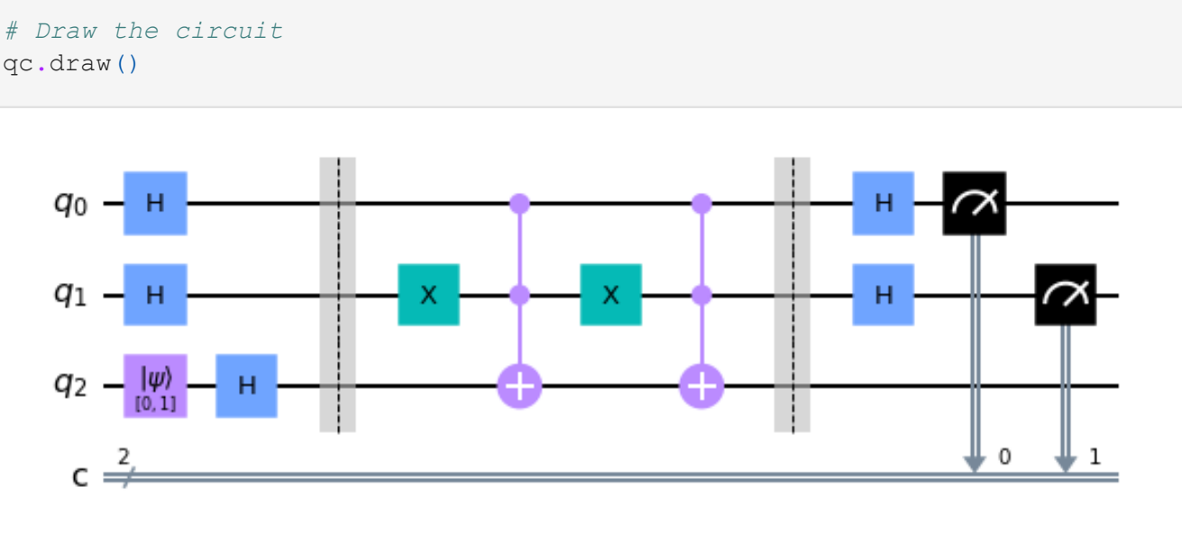
Task 3: For F_5

This is a balanced oracle thus we would expect that '00' won't appear with 100% certainity.

In [16]:

```
# Construct a circuit with a 3-qubit input, and 2-bit measurement storage.
qc = QuantumCircuit(3, 2)

# Initialise q2 with |1>
initial_state = [0, 1]
qc.initialize(initial_state, 2)

# Apply H-gates to q0, q1 and q2
qc.h(0)
qc.h(1)
qc.h(2)

# Insert the oracle F1
# And add two barriers to enclose this oracle.
qc.barrier()
qc.x(1)
qc.ccx(0,1,2)
qc.x(1)
qc.ccx(0,1,2)
qc.barrier()

# Apply H-gates to q0, q1
qc.h(0)
qc.h(1)

# Measure q0, q1
qc.measure([0,1], [0,1])

# Draw the circuit
qc.draw()
```

Out[16]:

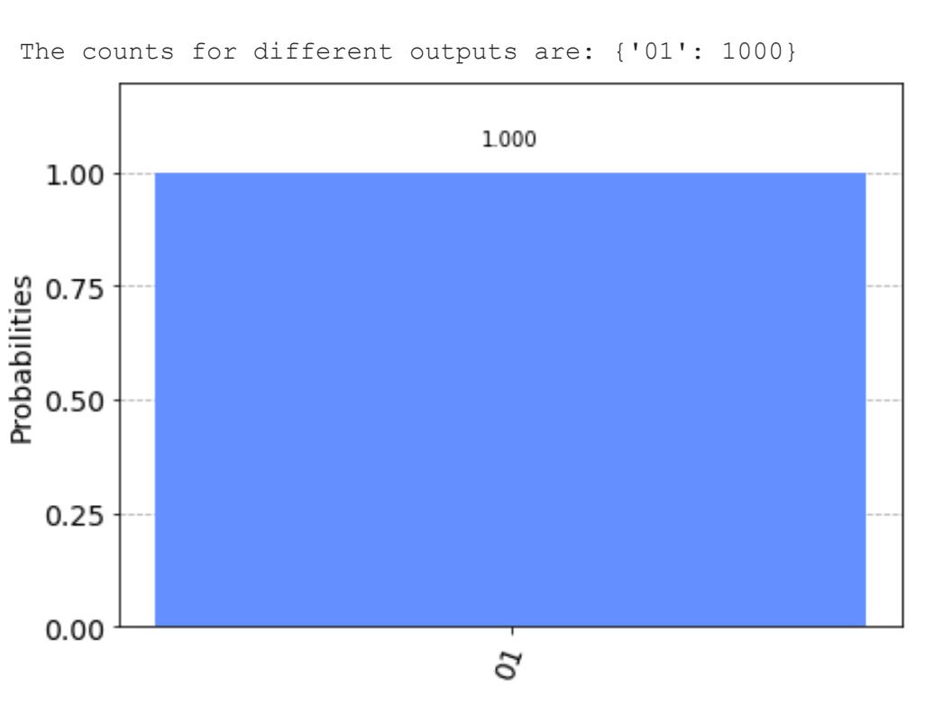


In [17]:

```
simulator = QasmSimulator()
# Run the circuit with certain number of shots.
# The 'shots' below means the number of times to run this circuit.
# Its default value is 1024, so if it is omitted, you'll run the circuit 1024 times.
job = execute(qc, backend=simulator, shots=1000)

# Call result() to obtain the result
result = job.result()

# Call get_counts() to obtain the counts of different outputs
counts = result.get_counts(qc)
# NB: 'counts' has a type of <class 'qiskit.result.counts.Counts'>,
# which is similar to a type of dictionary.
print(type(counts))

# Print the counts of different outputs
print("\n The counts for different outputs are:", counts)

# Plot the histogram
plot_histogram(counts)
```

```
<class 'qiskit.result.counts.Counts'>

 The counts for different outputs are: {'01': 1000}
```

Out[17]:



So we see that '00' indeed never appear. '10' and '11' never appear as well, which is particular to the oracle F_5. The calculation was done and it was confirmed that '01' is the expected result.