Practical 8

Name: Heja Bibani
Student Number: 16301173

In [8]:
```python
from qiskit import *
from qiskit.providers.aer import QasmSimulator
from qiskit.visualization import plot_histogram
%matplotlib inline
```

Section 1

Prove that the XOR operation satisfies Associativity: $(A \oplus B) \oplus C = A \oplus (B \oplus C)$

A	B	C	$(A \oplus B)$	$(B \oplus C)$	$(A \oplus B) \oplus C$	$A \oplus (B \oplus C)$
0	0	0	0	0	0	0
0	0	1	0	1	1	1
0	1	0	1	1	1	1
0	1	1	1	0	0	0
1	0	0	1	0	1	1
1	0	1	1	1	0	0
1	1	0	0	1	0	0
1	1	1	0	0	1	1

In the table above we notice that the output of both tables are the same and thus satisfies associativity $(A \oplus B) \oplus C = A \oplus (B \oplus C)$.

Section 2

Task 2.1

Derive the gate matrix for the oracle F_3.

We know the following for $F_3(x)$:

$F_3(0)$ = 1

$F_3(1)$ = 1

We generate the following table, this would consequently act as a not gate on the y-qubit, since we are using XOR with 1:

$/y\rangle, /x\rangle$	$/y \oplus F_3(x)\rangle, /x\rangle$	**Output**
$\|0\rangle, \|0\rangle$	$\|0 \oplus F_3(0)\rangle, \|0\rangle$	$\|1\rangle, \|0\rangle$
$\|0\rangle, \|1\rangle$	$\|0 \oplus F_3(1)\rangle, \|1\rangle$	$\|1\rangle, \|1\rangle$
$\|1\rangle, \|0\rangle$	$\|1 \oplus F_3(0)\rangle, \|0\rangle$	$\|0\rangle, \|0\rangle$
$\|1\rangle, \|1\rangle$	$\|1 \oplus F_3(1)\rangle, \|1\rangle$	$\|0\rangle, \|1\rangle$

The gate matrix would be of the following:

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

Task 2.2

Show this gate matrix is its own inverse.

The gate is of the following:

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

Multiplying the gate with its self produces the identity element.

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

Thus it is its own inverse.

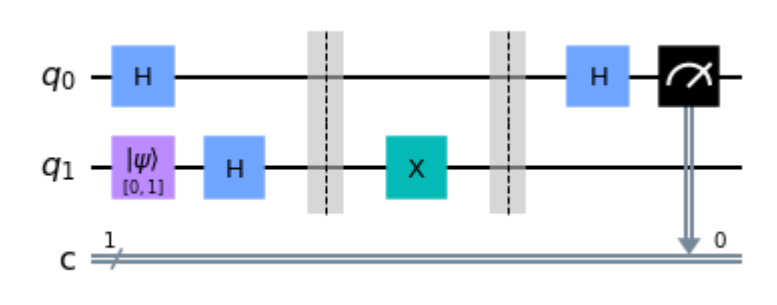
Task 2.3

Use a Quantum Circuit to implement the oracle F_3, and draw this circuit.

In this case, the circuit will be the same as applying a not-gate on $|y\rangle$. Applying this gate gives the exact outputs given in section 2.1. We are expecting that the value will be guaranteed to be 0.

In [9]:
```python
# Construct a circuit with a 2-qubit input, and 1-bit measurement storage.
qc = QuantumCircuit(2, 1)

# Initialise q1 with |1>
initial_state = [0, 1]
qc.initialize(initial_state, 1)

# Apply H-gates to both q0 and q1
qc.h(0)
qc.h(1)
# Insert the oracle, which is F0 (the I gate)
# Since I gate does nothing, we simply add two barriers here.
qc.barrier()
qc.x(1)
qc.barrier()
# Apply H-gate to q0
qc.h(0)
# Measure q0
qc.measure(0, 0)

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

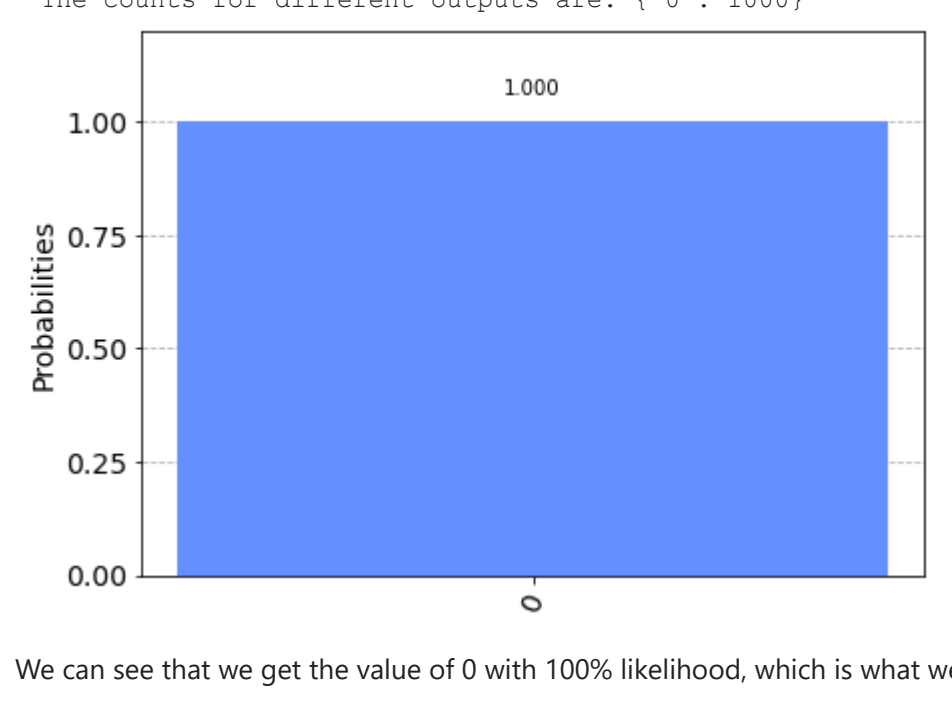
Out[9]:



In [10]:
```python
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: {'0': 1000}
```

Out[10]:



We can see that we get the value of 0 with 100% likelihood, which is what we have expected.

Section 3

Use the QasmSimulator to implement the following variant of the Deutsch Algorithm: The circuit should be the same as described in our lecture, but both of the input qubits take the state of $|1\rangle$.

Task 3.1

Show by calculation that this variant can also tell if an oracle is constant or balanced by one run of the circuit. Especially, please state what measurement outcome of the top qubit indicates a constant oracle and what measurement outcome indicates a balanced oracle.

When we set the values to both $|1\rangle$. Then both will be in the following state:

$\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$

Thus joining the two gates, we must get the tensor product:

$= (\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle) \otimes (\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle)$

$= \frac{1}{2}|00\rangle - \frac{1}{2}|01\rangle - \frac{1}{2}|10\rangle + \frac{1}{2}|11\rangle$

This is the same as:

$= \frac{1}{2}|0\rangle \otimes |0\rangle - \frac{1}{2}|0\rangle \otimes |1\rangle - \frac{1}{2}|1\rangle \otimes |0\rangle + \frac{1}{2}|1\rangle \otimes |1\rangle$

After it passes through the Oracle it becomes:

$= \frac{1}{2}|0 \oplus f_i(0)\rangle \otimes |0\rangle - \frac{1}{2}|0 \oplus f_i(1)\rangle \otimes |1\rangle - \frac{1}{2}|1 \oplus f_i(0)\rangle \otimes |0\rangle + \frac{1}{2}|1 \oplus f_i(1)\rangle \otimes |1\rangle$

We will then combine the results:

$= \frac{1}{2}(|0 \oplus f_i(0)\rangle - |1 \oplus f_i(0)\rangle) \otimes |0\rangle - \frac{1}{2}(|0 \oplus f_i(1) - |1 \oplus f_i(1)\rangle) \otimes |1\rangle$

We note that $|0 \oplus f_i(0)\rangle$ and $|1 \oplus f_i(0)\rangle$ will always have one being $|0\rangle$ and the other being $|1\rangle$, so we have the following:

if $f_i(0)$ = 0, then $|0 \oplus f_i(0)\rangle - |1 \oplus f_i(0)\rangle = |0\rangle - |1\rangle$

if $f_i(0)$ = 1, then $|0 \oplus f_i(0)\rangle - |1 \oplus f_i(0)\rangle = |1\rangle - |0\rangle = -(|0\rangle - |1\rangle)$

Combining the above:

$|0 \oplus f_i(0)\rangle - |1 \oplus f_i(0)\rangle = (-1)^{f_i(0)}(|0\rangle - |1\rangle)$

$|0 \oplus f_i(1)\rangle - |1 \oplus f_i(1)\rangle = (-1)^{f_i(1)}(|0\rangle - |1\rangle)$

Thus:

$\frac{1}{2}(|0 \oplus f_i(0)\rangle - |1 \oplus f_i(0)\rangle) \otimes |0\rangle - \frac{1}{2}(|0 \oplus f_i(1) - |1 \oplus f_i(1)\rangle) \otimes |1\rangle$

$= \frac{1}{2}(-1)^{f_i(0)}(|0\rangle - |1\rangle) \otimes |0\rangle - \frac{1}{2}(-1)^{f_i(1)}(|0\rangle - |1\rangle) \otimes |1\rangle$

$= \frac{1}{2}(|0\rangle - |1\rangle) \otimes (-1)^{f_i(0)}|0\rangle - \frac{1}{2}(|0\rangle - |1\rangle) \otimes (-1)^{f_i(1)}|1\rangle$

$= \frac{1}{2}(|0\rangle - |1\rangle) \otimes ((-1)^{f_i(0)}|0\rangle - (-1)^{f_i(1)}|1\rangle)$

$= \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \otimes \frac{1}{\sqrt{2}}((-1)^{f_i(0)}|0\rangle - (-1)^{f_i(1)}|1\rangle)$

It is clear that the top wire is:

$\frac{1}{\sqrt{2}}((-1)^{f_i(0)}|0\rangle - (-1)^{f_i(1)}|1\rangle)$

From this we can see that it will take the following values:

For f_0, $f_0(0)$ = 0 and $f_0(1)$ = 0 it equals $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$

For f_1, $f_1(0)$ = 0 and $f_1(1)$ = 1 it equals $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$

For f_2, $f_2(0)$ = 1 and $f_2(1)$ = 0 it equals $\frac{1}{\sqrt{2}}(-|0\rangle - |1\rangle)$ = $-\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$

For f_3, $f_3(0)$ = 1 and $f_3(1)$ = 1 it equals $\frac{1}{\sqrt{2}}(-|0\rangle + |1\rangle)$ = $-\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$

So if we apply the H-gate to the top qubit we'll get $|0\rangle$ or $|1\rangle$

For f_0, $H(\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)) = |1\rangle$

For f_1, $H(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)) = |0\rangle$

For f_2, $H(-\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)) = -|0\rangle$

For f_3, $H(-\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)) = -|1\rangle$

This shows that the value for 1 indicates a constant oracle and conversely a value of 0 indicates a balanced oracle.

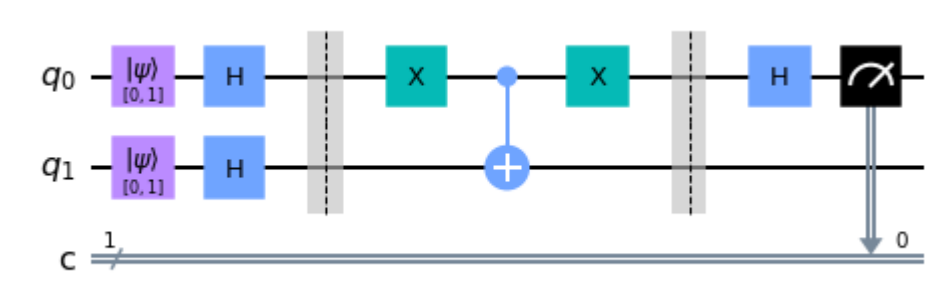
Task 3.2

Implement this variant with F_2 as the oracle.

As expected in the above, applying F_2, we would expect the value of the top qubit to be 0 with 100% probability, as this is a balanced oracle. We apply the gate for F_2 as a variant of the CNOT gate as described in the lecture by applying a NOT gate to the top qubit and then restoring its state after applying the CNOT gate. We have also initialized the qubits to the state $|1\rangle$ as required.

In [11]:
```python
# Construct a circuit with a 2-qubit input, and 1-bit measurement storage.
qc = QuantumCircuit(2, 1)

# Initialise q1 with |1>
initial_state = [0, 1]
qc.initialize(initial_state, 1)
qc.initialize(initial_state, 0)
# Apply H-gates to both q0 and q1
qc.h(0)
qc.h(1)
# Insert the oracle, which is F0 (the I gate)
# Since I gate does nothing, we simply add two barriers here.
qc.barrier()
qc.x(0)
qc.cx(0,1)
qc.x(0)
qc.barrier()
# Apply H-gate to q0
qc.h(0)
# Measure q0
qc.measure(0, 0)

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

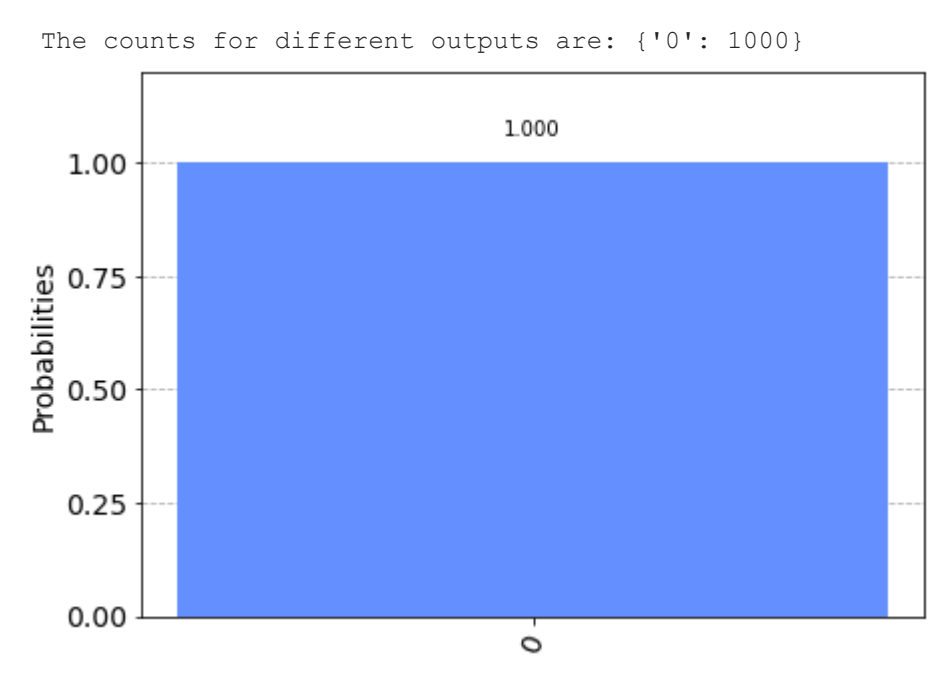
Out[11]:



In [12]:
```python
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: {'0': 1000}
```

Out[12]:



As expected the value of the bit is 0 with 100% certainity.

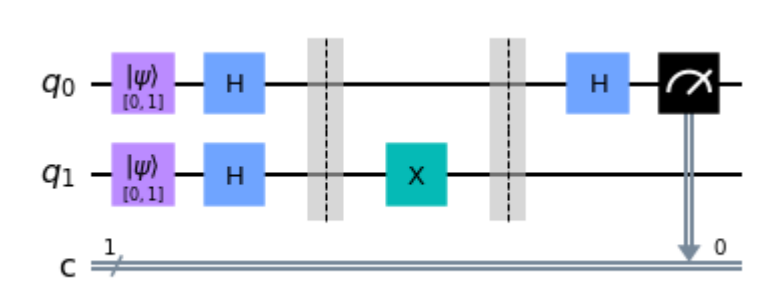
Task 3.3

Implement this variant with F_3 as the oracle.

As expected in the above, applying F_3, we would expect the value of the top qubit to be 1 with 100% probability, as this is a constant oracle. We apply the gate for F_3 as applying a NOT gate on the y-input. We have also initialized the qubits to the state $|1\rangle$ as required.

In [13]:
```python
# Construct a circuit with a 2-qubit input, and 1-bit measurement storage.
qc = QuantumCircuit(2, 1)

# Initialise q1 with |1>
initial_state = [0, 1]
qc.initialize(initial_state, 1)
qc.initialize(initial_state, 0)
# Apply H-gates to both q0 and q1
qc.h(0)
qc.h(1)
# Insert the oracle, which is F0 (the I gate)
# Since I gate does nothing, we simply add two barriers here.
qc.barrier()
qc.x(1)
qc.barrier()
# Apply H-gate to q0
qc.h(0)
# Measure q0
qc.measure(0, 0)

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

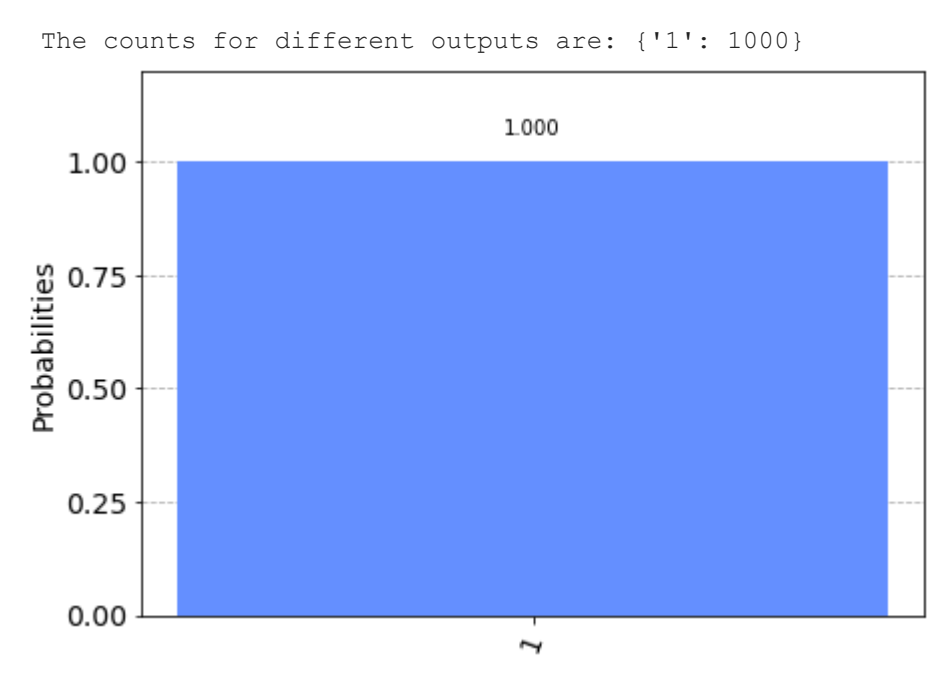
Out[13]:



In [14]:
```python
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: {'1': 1000}
```

Out[14]:



As expected the value of the bit is 1 with 100% certainity.