```
In [1]: %run utils.py
```

COMS 4281 Intro to Quantum Computing

Problem Set 1: Quantum Info Basics

Due: Sept [illegible] 9pm

Collaboration [illegible] aged (teams of at most 3). Please read the syllabus carefully for th[illegible] collaboration. In particular, everyone must write their own solutions [illegible]

Please write your collaborators here:

Problem 1: Bra-Ket Notation Basics

Define $|\psi\rangle = \alpha|0\rangle + \beta|+\rangle$.

1. Write out the state $|\psi\rangle$ as linear combinations of the standard basis states $|0\rangle, |1\rangle$. In order for $|\psi\rangle$ to be a valid quantum state (i.e. be a unit vector), what are the conditions on α, β?
2. Since $\{|+\rangle, |-\rangle\}$ also forms a basis of $\mathbb{C}^2$ (sometimes called the *diagonal basis*), write $|\psi\rangle$ as a linear combinations of $|+\rangle$ and $|-\rangle$.
3. What is the probability of obtaining the $|0\rangle$ and $|1\rangle$ states when measuring $H|\psi\rangle$?

Solution

write your solution here, using LaTeX and Markdown

Problem 2: Outer Products and Projections

Recall that $\langle\psi|\theta\rangle$ denotes a scalar, because is an inner product between two vectors (i.e., you have a row vector followed by a column vector). Now consider $|\psi\rangle\langle\theta|$, which denotes the outer product between the two vectors (i.e., a column vector followed by a row vector). The outer product of two vectors is a matrix.

1. Show that $|0\rangle\langle 0| = \frac{1}{2}(I+Z)$ and $|+\rangle\langle +| = \frac{1}{2}(I+X)$, where $Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$ and $X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$.

2. Let $\{|0\rangle, |1\rangle, \ldots, |n-1\rangle\}$ denote the standard basis for $\mathbb{C}^n$. Show that $I = \sum_{x=0}^{n-1} |x\rangle\langle x|$. This simple identity known as the `completeness relation` can be very us[illegible]

3. Use the c[illegible] to show that $|\psi\rangle = \sum_x \langle x|\psi\rangle |x\rangle$ for any vector $|\psi\rangle$. Thus the [illegible] the standard basis, can be written as inner products of $|\psi\rangle$ with t[illegible]ctors.

Solution

write your solution here, using LaTeX and Markdown

Problem 3: Composite Quantum Systems

Here are some questions to help you get used to the tensor product.

Let $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{m\times m}$. Recall that the tensor product $A \otimes B$ can be given by:

$$A \otimes B = \begin{bmatrix} A_{11}B & A_{12}B & \cdots & A_{1n}B \\ A_{11}B & A_{12}B & \cdots & A_{2n}B \\ \vdots & \vdots & \ddots & \vdots \\ A_{n1}B & A_{n2}B & \cdots & A_{nn}B \end{bmatrix}$$

is an $nm \times nm$ dimensional matrix. This is also known as the *Kronecker product* which is a way to represent the tensor product with respect to a given basis.

1. Give the explicit matrix representations for the matrices $I \otimes H$, $H \otimes I$ and $H \otimes H$.

2. Compute $(I \otimes H)|0,1\rangle$, $(H \otimes I)|0,1\rangle$ and $(H \otimes H)|0,1\rangle$, expressing the results in the standard basis (i.e. $\sum_{i,j\in\{0,1\}} c_{ij}|i\rangle|j\rangle$ and some constants $c_{ij} \in \mathbb{C}$)

Solution

write your solution here, using LaTeX and Markdown

Problem 4: Quantum Entanglement

For each of the following quantum states, say whether they are entangled or unentangled across the specified bipartition, and justify your answer.

1. $|\Phi\rangle = \frac{1}{\sqrt{2}}\Big(|00\rangle + |11\rangle\Big) = \begin{pmatrix} 1/\sqrt{2} \\ 0 \\ 0 \\ 1/\sqrt{2} \end{pmatrix}$ is a two-qubit state in $\mathbb{C}^2 \otimes \mathbb{C}^2$, called the `EPR pai` [illegible] s entangled?

2. $|\psi\rangle = \frac{1}{\sqrt{2}}$ [illegible] n-qubit state where $|x\rangle = |x_1\rangle \otimes |x_2\rangle \otimes \cdots \otimes |x_n\rangle$ is a basis stat [illegible] y we divide the n qubits into the first j and the last $n - j$ qub [illegible] ngled across the bipartition $(\mathbb{C}^2)^{\otimes j} \otimes (\mathbb{C}^2)^{\otimes (n-j)}$?

3. Let $|\Phi\rangle$ be the EPR pair. Let $H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ denote the Hadamard gate. Is the state $(I \otimes H)\,|\Phi\rangle$ entangled?

4. Is the state $CNOT\,|\Phi\rangle$ entangled?

Solution

write your solution here, using LaTeX and Markdown

Problem 5: Implement a Quantum Circuit

In this problem, you will implement a simple quantum circuit that constructs the $|\psi\rangle = \frac{1}{\sqrt{2}}(|000\rangle - |111\rangle)$ from the all zeroes state. In other words, you will find a circuit C such that

$$C\,|000\rangle = |\psi\rangle$$

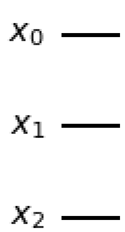
In this problem, you will use the Qiskit library to implement, visualize, and analyze the circuit C.

1. Design a circuit C to prepare the state $|\psi\rangle$, and write the corresponding Qiskit code between the "BEGIN CODE" and "END CODE" delineations below. You may use any of the gates we have learned in class. We've already created the circuit object, you just need to specify what gates to add.

In [2]:
```
def create_sym_state_circuit():
    qr = QuantumRegister(3, name='x')
    qc = QuantumCircuit(qr)
```

```
    # ========= BEGIN CODE =================


    # ========= END CODE =================


    return c

qc = create_                        ()
qc.draw(outp
```


Out[2]:



3. Consult the qiskit documentation and use its API to obtain the output state of the circuit you just created as a **vector** (i.e. a list of amplitudes).
   It doesn't matter for this problem, but keep in mind that qiskit uses the **little-endian** convention (e.g., qubits are ordered right to left; for more info see this). We provide a function *beautify*, which given input `amplitude` and `k` where `amplitude` is a list of 2^k complex numbers and k is an integer, prints the amplitude list as a quantum state. See the code below as an example.

In [16]:
```
# Example for beautify
amplitudes = [complex(-1.55345, 0),   complex(0,0),   complex(-1, 0.1),    comp
print('State vector:', beautify(amplitudes, 2))
```

```
State vector: -1.55 |00> + (-1+0.1i) |01> - 1.1i |11>
```

In the following code block, print the output of the circuit you created in Part 1 above as a vector in beautified form.

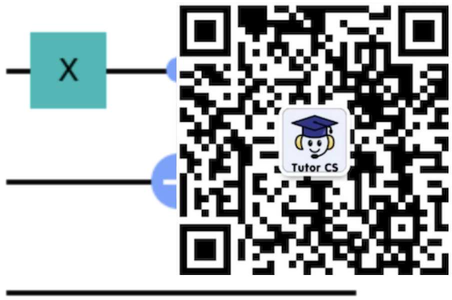
In [4]:
```
# ========= BEGIN CODE =================


# ========= END CODE =================
```

4. Write code to measure all the qubits of the $|\psi\rangle$ state in the standard basis and visualize the measurement statistics using a histogram.

In [5]:
```
# ========= BEGIN CODE =================
```

```
# ========= END CODE ================
```


5. Consider running the following circuit with $|\psi\rangle$ as input. Let $|\theta\rangle$ denote the output state. Calculate the state $|\theta\rangle$ by computing the intermediate states of the circuit, and write it out below in $\LaTeX$.



Solution

6. Write code to implement a circuit that prepares the state $|\theta\rangle$, measure it in the standard basis and visualize the statistics using a histogram.

In [6]:
```
# ========= BEGIN CODE ================

# ========= END CODE ================
```


Problem 6: A Quantum Two-bit Adder

The classical two-bit adder is an **irreversible** function that takes in four bits, (A_0, A_1) and (B_0, B_1), and outputs three bits (C_0, Q_0, Q_1) which is the binary representation of the sum of $2A_1 + A_0$ and $2B_1 + B_0$ (i.e., integers that (A_0, A_1) and (B_0, B_1) represent in binary). For example, on input $(0, 1)$ and $(1, 1)$ the two bit adder should return $(1, 0, 0)$. On input $(1, 1)$ and $(1, 1)$ it should output $(1, 1, 0)$.

You can find a circuit for an irreversible circuit for the two-bit adder here, consisting of XOR, OR, and AND gates (see Wikipedia for gate symbol reference).

In this problem you will implement a **reversible** two-bit adder in Qiskit.

1. First, let's implement reversible versions of the XOR, OR, and AND gates. Recall that every boolean function f can be converted to a reversible transformation T_f using an additional ancilla bit. Since XOR, OR, AND map 2 bits to 1 bit, the reversible functions T_{XOR}, T_{OR}, T_{AND} will map 3 bits to 3 bits. The corresponding matrices are 8×8.

In the functions below, enter the matrix representations of T_{XOR}, T_{OR}, T_{AND} below (replace the entries with the appropriate values). The row/columns are ordered as follows: $|000\rangle, |001\rangle, |010\rangle, \ldots, |111\rangle$.

Your implementations of reversible XOR, OR, and AND will be tested.

In [7]:

```
def create_Tor(qr: QuantumRegister) -> QuantumCircuit:
    assert l[illegible] gate should operate on 3 qubits.'
    qc = Qua[illegible]
    ##### FI[illegible]BELOW FOR THE REVERSIBLE OR GATE ##########
    Tor = Op[illegible]
        [1, [illegible] 0],
        [0, [illegible] 0],
        [0, [illegible] 0],
        [0, [illegible] 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],
    ])
    ####################################################################
    qc.unitary(Tor, [2, 1, 0], label='Tor')
    return qc

def create_Txor(qr: QuantumRegister) -> QuantumCircuit:
    assert len(qr) == 3, 'Txor gate should operate on 3 qubits.'
    qc = QuantumCircuit(qr)
    ##### FILL IN THE MATRIX BELOW FOR THE REVERSIBLE XOR GATE ##########
    Txor = Operator([
        [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],
    ])
    ####################################################################
    qc.unitary(Txor, [2, 1, 0], label='Txor')
    return qc

def create_Tand(qr: QuantumRegister) -> QuantumCircuit:
    assert len(qr) == 3, 'Tand gate should operate on 3 qubits.'
    qc = QuantumCircuit(qr)
    ##### FILL IN THE MATRIX BELOW FOR THE REVERSIBLE AND GATE ##########
    Tand = Operator([
        [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],
    ])
```

```
    ########################################################################
    qc.unitary(Tand, [2, 1, 0], label='Tand')
    return qc
```

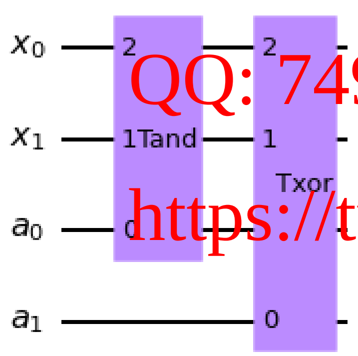

In [8]:
```
#Running test cases on your adder...
test_gates([create_Tor,create_Txor,create_Tand])
```

```
Testing gates...
OR gate: Er
XOR gate: E
AND gate: E
```

3. You can n rsible circuits consisting of T_{XOR}, T_{OR}, and T_{AND} by using the or, `create_Txor`, and `create_Tand`, and also a helper fur that allows you to append a gate G to a circuit C. The function takes in a circuit C, a function g constructs the gate G, and a list of bits that G operates on. See the code below as an example.

In [9]:
```
### EXAMPLE ONLY ###############
qx = QuantumRegister(2, name="x")
qa = QuantumRegister(2, name="a")
qc = QuantumCircuit(qx,qa)                #add the X register and A registers
qc = append(qc, create_Tand, [0, 1, 2])   #reversible AND acting on (x0,x1) anc
qc = append(qc, create_Txor, [0,1,3])     #reversible XOR acting on (x0,x1) anc
qc.draw(output='mpl')
```


Out[9]:



Now, transform the irreversible circuit for the two-bit adder above to a **reversible** circuit C for the two-bit adder. More precisely, the circuit C should act on bits

- (A_0, A_1) representing the first number $A = 2A_1 + A_0$
- (B_0, B_1) representing the second number $B = 2B_1 + B_0$
- (C_0, C_1, C_2) representing the binary representation of $A + B$
- Some number of ancilla bits $(D_0, D_1, \ldots)$

The circuit C should have the behavior: for all inputs $A_0, A_1, B_0, B_1 \in \{0, 1\}$,

$$C\,|A, B, 0, 0 \cdots 0\rangle = \Big|\underbrace{A}_{\text{2 bits}}, \underbrace{B}_{\text{2 bits}}, \underbrace{A+B}_{\text{3 bits}}, \underbrace{S_{A,B}}_{\text{ancillas}}\Big\rangle$$

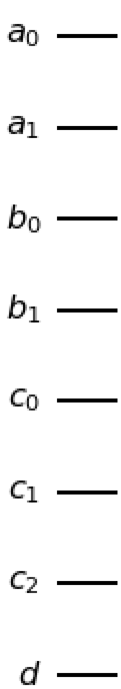
where A, B are two bits and $A + B$ is represented by three bits. $S_{A,B}$ corresponds to the bits of the ancilla that depends on the inputs A, B. This data corresponds to the "scratch work" of the computation.

Your circuit C can use T_{XOR}, T_{AND}, T_{OR}, as well as $CNOT$, X, Z and H gates. Choose the appropriate number of ancillas, and then implement your circuit where indicated. The code afterwards will visualize your circuit as well run it on several test cases.

In [10]:

```
# Todo: fil[illegible] ancillary qubits for your circuit
num_anc = 1

def create_two_bit_adder_with_scratch(num_anc):
    A = Quar[illegible]me="a")
    B = Quar[illegible]me="b")
    C = QuantumRegister(3, name="c")
    D = QuantumRegister(num_anc, name="d")
    qc = QuantumCircuit(A,B,C,D)

    #### BEGIN YOUR CODE HERE ############################



    ##### END YOUR CODE HERE ################################

    return qc

two_bit_adder_with_scratch = create_two_bit_adder_with_scratch(num_anc=num_anc)
two_bit_adder_with_scratch.draw(output='mpl')
```


Out[10]:



In [11]:

```
# Running test cases on your adder....
test_two_bit_adder(two_bit_adder_with_scratch, num_anc, has_scratch=True)
```

```
Testing two-bit adder with scratch...
Error (incorrect): A = 00, B = 01, C = 000.
Error (incorrect): A = 00, B = 10, C = 000.
Error (incorrect): A = 00, B = 11, C = 000.
Error (incorrect): A = 01, B = 00, C = 000.
Error (incorrect): A = 01, B = 01, C = 000.
Error (incorrect): A = 01, B = 10, C = 000.
Error (incorrect): A = 01, B = 11, C = 000.
Error (incorrect): A = 10, B = 00, C = 000.
Error (incorrect): A = 10, B = 01, C = 000.
Error (incorrect): A = 10, B = 10, C = 000.
Error (incorrect): A = 10, B = 11, C = 000.
Error (incorrect): A = 11, B = 00, C = 000.
Error (incorrect): A = 11, B = 01, C = 000.
Error (incorrect): A = 11, B = 10, C = 000.
Error (incorrect): A = 11, B = 11, C = 000.
```


4. Now we go one step further to implement a reversible two-bit adder that does the same thing as above except the scratch bits start **and end** in the zero state.

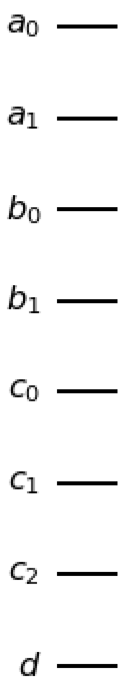
$$C\,|A, B, 0, 0\cdots 0\rangle = |A, B, A+B, 0\cdots 0\rangle$$

In other words, the scratch work is erased.

Hint: Use an additional ancilla to save the output, and then reverse the computation.

In [12]:
```python
def create_two_bit_adder() -> QuantumCircuit:
    A = QuantumRegister(2, name="a")
    B = QuantumRegister(2, name="b")
    C = QuantumRegister(3, name="c")
    D = QuantumRegister(num_anc, name="d")
    qc = QuantumCircuit(A,B,C,D)

    #### BEGIN YOUR CODE HERE ##############################



    ##### END YOUR CODE HERE ###############################

    return qc

two_bit_adder = create_two_bit_adder()
two_bit_adder.draw(output='mpl')
```

Out[12]:



In [13]:

```
#Running test cases on your adder....
test_two_bit_adder(two_bit_adder, num_anc, has_scratch=False)
```

```
Testing two-bit adder without scratch...
Error (incorrect): A = 00, B = 01, C = 000.
Error (incorrect): A = 00, B = 10, C = 000.
Error (incorrect): A = 00, B = 11, C = 000.
Error (incorrect): A = 01, B = 00, C = 000.
Error (incorrect): A = 01, B = 01, C = 000.
Error (incorrect): A = 01, B = 10, C = 000.
Error (incorrect): A = 01, B = 11, C = 000.
Error (incorrect): A = 10, B = 00, C = 000.
Error (incorrect): A = 10, B = 01, C = 000.
Error (incorrect): A = 10, B = 10, C = 000.
Error (incorrect): A = 10, B = 11, C = 000.
Error (incorrect): A = 11, B = 00, C = 000.
Error (incorrect): A = 11, B = 01, C = 000.
Error (incorrect): A = 11, B = 10, C = 000.
Error (incorrect): A = 11, B = 11, C = 000.
```

In [ ]: