Qiskit Dev Certificate Sample Questions

Inho Choi

Qiskit Advocate

Qiskit

Question 1

Qiskit

1. **Which statement will create a quantum circuit with four quantum bits and four classical bits?**

A. `QuantumCircuit(4, 4)`
B. `QuantumCircuit(4)`
C. `QuantumCircuit(QuantumRegister(4, 'qr0'), QuantumRegister(4, 'cr1'))`
D. `QuantumCircuit([4, 4])`

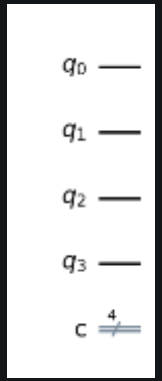
Question 1

Qiskit

1. **Which statement will create a quantum circuit with four quantum bits and four classical bits?**

   A. `QuantumCircuit(4, 4)`
   B. `QuantumCircuit(4)`
   C. `QuantumCircuit(QuantumRegister(4, 'qr0'), QuantumRegister(4, 'cr1'))`
   D. `QuantumCircuit([4, 4])`

A.



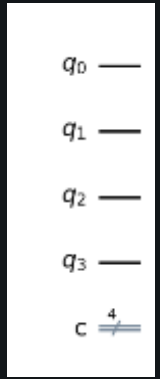
Question 1

Qiskit

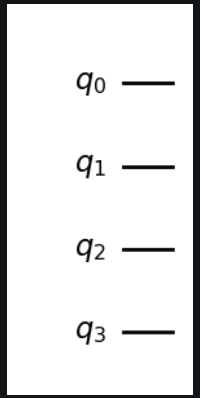
1. **Which statement will create a quantum circuit with four quantum bits and four classical bits?**

   A. `QuantumCircuit(4, 4)`
   B. `QuantumCircuit(4)`
   C. `QuantumCircuit(QuantumRegister(4, 'qr0'), QuantumRegister(4, 'cr1'))`
   D. `QuantumCircuit([4, 4])`

A.



B.



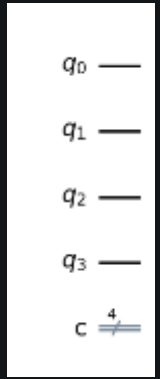
Question 1

Qiskit

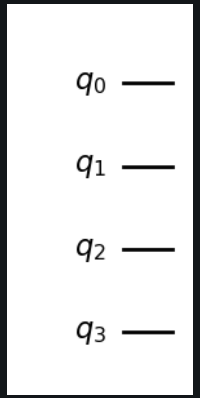
1. **Which statement will create a quantum circuit with four quantum bits and four classical bits?**

   A. `QuantumCircuit(4, 4)`
   B. `QuantumCircuit(4)`
   C. `QuantumCircuit(QuantumRegister(4, 'qr0'), QuantumRegister(4, 'cr1'))`
   D. `QuantumCircuit([4, 4])`

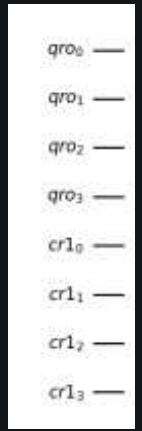
A.



B.



C.



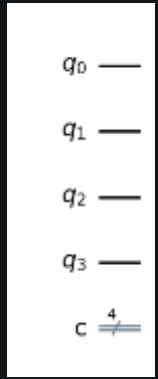
Question 1

Qiskit

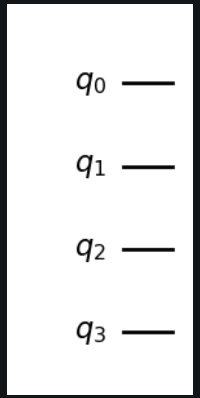
1. **Which statement will create a quantum circuit with four quantum bits and four classical bits?**

```
A. QuantumCircuit(4, 4)
B. QuantumCircuit(4)
C. QuantumCircuit(QuantumRegister(4, 'qr0'),
   QuantumRegister(4, 'cr1'))
D. QuantumCircuit([4, 4])
```

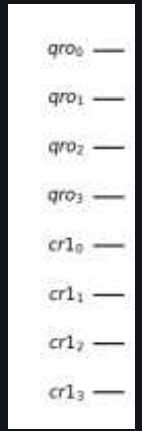
A.



B.



C.



D.

```
Error: 'Expected an instance of Qubit, Clbit, or AncillaQubit, but was passed 4'
```

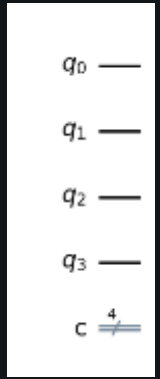
Question 1

Qiskit

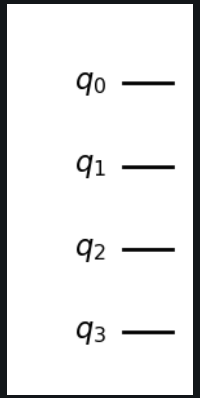
1. **Which statement will create a quantum circuit with four quantum bits and four classical bits?**

   A. `QuantumCircuit(4, 4)`
   B. `QuantumCircuit(4)`
   C. `QuantumCircuit(QuantumRegister(4, 'qr0'), QuantumRegister(4, 'cr1'))`
   D. `QuantumCircuit([4, 4])`

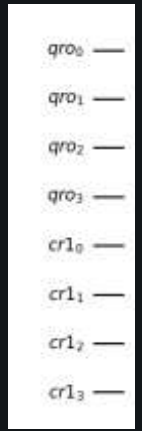
A.



B.



C.



D.

```
Error: 'Expected an instance of Qubit, Clbit, or AncillaQubit, but was passed 4'
```

Question 2

Qiskit

2. Given this code fragment, what is the probability that a measurement would result in |0> ?

```
qc = QuantumCircuit(1)
qc.ry(3 * math.pi/4, 0)
```

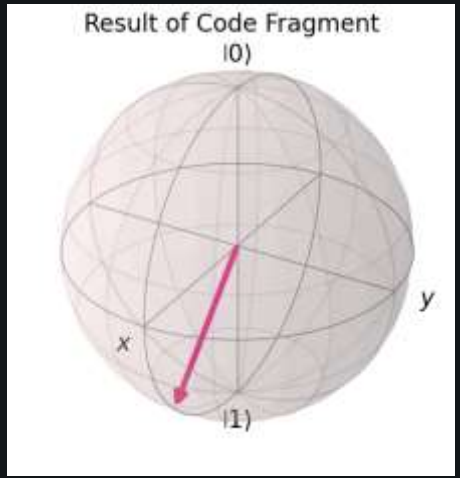
A. 0.8536
B. 0.5
C. 0.1464
D. 1.0

Question 2

2. Given this code fragment, what is the probability that a measurement would result in |0> ?

```
qc = QuantumCircuit(1)
qc.ry(3 * math.pi/4, 0)
```

A. 0.8536
B. 0.5
C. 0.1464
D. 1.0

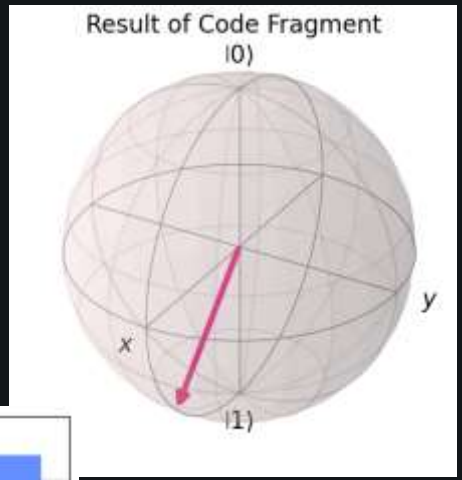


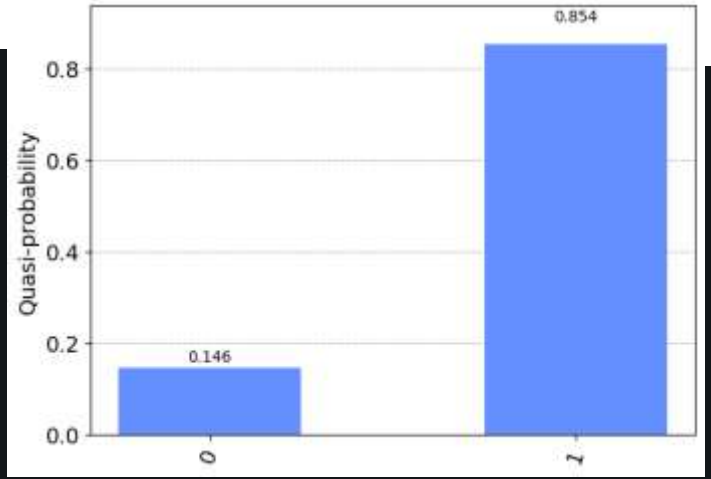
Question 2

2. Given this code fragment, what is the probability that a measurement would result in |0> ?

```
qc = QuantumCircuit(1)
qc.ry(3 * math.pi/4, 0)
```

A. 0.8536
B. 0.5
C. 0.1464
D. 1.0



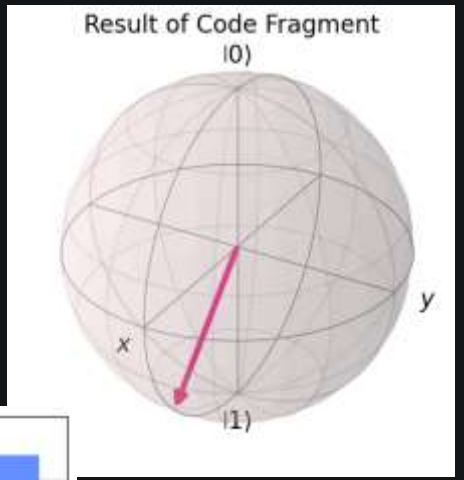


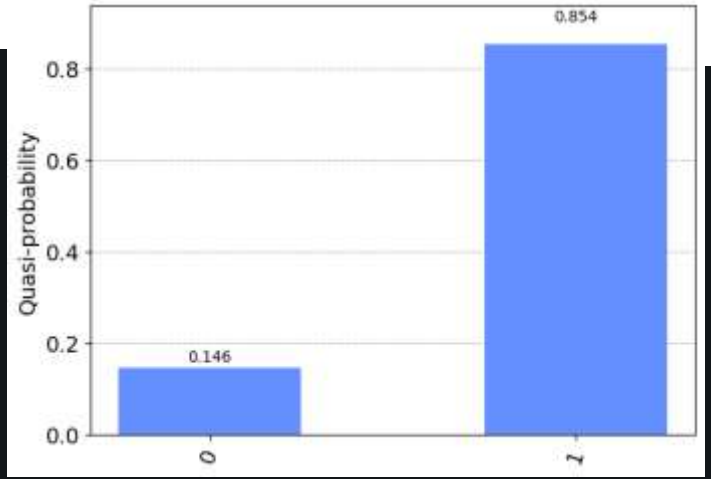
Question 2

2. Given this code fragment, what is the probability that a measurement would result in |0> ?

```
qc = QuantumCircuit(1)
qc.ry(3 * math.pi/4, 0)
```

A. 0.8536
B. 0.5
C. 0.1464
D. 1.0

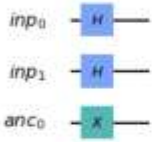




Question 3

3. Assuming the fragment below, which three code fragments would produce the circuit illustrated?

```
inp_reg = QuantumRegister(2, name='inp')
ancilla = QuantumRegister(1, name='anc')
qc = QuantumCircuit(inp_reg, ancilla)

# Insert code here
```

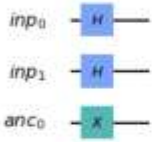


A.
```
qc.h(inp_reg)
qc.x(ancilla)
qc.draw()
```
B.
```
qc.h(inp_reg[0:2])
qc.x(ancilla[0])
qc.draw()
```
C.
```
qc.h(inp_reg[0:1])
qc.x(ancilla[0])
qc.draw()
```
D.
```
qc.h(inp_reg[0])
qc.h(inp_reg[1])
qc.x(ancilla[0])
qc.draw()
```
E.
```
qc.h(inp_reg[1])
qc.h(inp_reg[2])
qc.x(ancilla[1])
qc.draw()
```
F.
```
qc.h(inp_reg)
qc.h(inp_reg)
qc.x(ancilla)
qc.draw()
```

Question 3

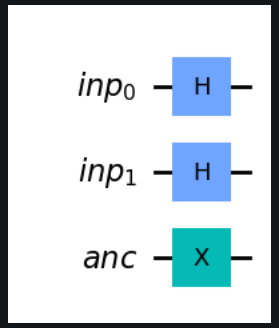
3. Assuming the fragment below, which three code fragments would produce the circuit illustrated?

```
inp_reg = QuantumRegister(2, name='inp')
ancilla = QuantumRegister(1, name='anc')
qc = QuantumCircuit(inp_reg, ancilla)

# Insert code here
```



A.
```
qc.h(inp_reg)
qc.x(ancilla)
qc.draw()
```
B.
```
qc.h(inp_reg[0:2])
qc.x(ancilla[0])
qc.draw()
```
C.
```
qc.h(inp_reg[0:1])
qc.x(ancilla[0])
qc.draw()
```
D.
```
qc.h(inp_reg[0])
qc.h(inp_reg[1])
qc.x(ancilla[0])
qc.draw()
```
E.
```
qc.h(inp_reg[1])
qc.h(inp_reg[2])
qc.x(ancilla[1])
qc.draw()
```
F.
```
qc.h(inp_reg)
qc.h(inp_reg)
qc.x(ancilla)
qc.draw()
```

A.

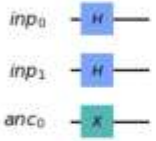


Qiskit

Question 3

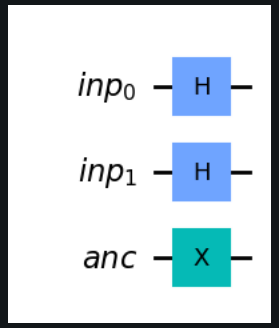
3. Assuming the fragment below, which three code fragments would produce the circuit illustrated?

```
inp_reg = QuantumRegister(2, name='inp')
ancilla = QuantumRegister(1, name='anc')
qc = QuantumCircuit(inp_reg, ancilla)

# Insert code here
```

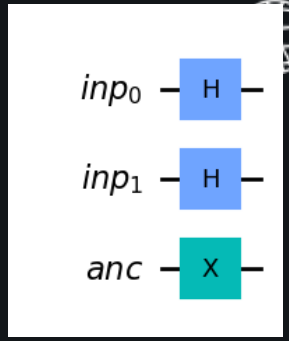


A.
```
qc.h(inp_reg)
qc.x(ancilla)
qc.draw()
```
B.
```
qc.h(inp_reg[0:2])
qc.x(ancilla[0])
qc.draw()
```
C.
```
qc.h(inp_reg[0:1])
qc.x(ancilla[0])
qc.draw()
```
D.
```
qc.h(inp_reg[0])
qc.h(inp_reg[1])
qc.x(ancilla[0])
qc.draw()
```
E.
```
qc.h(inp_reg[1])
qc.h(inp_reg[2])
qc.x(ancilla[1])
qc.draw()
```
F.
```
qc.h(inp_reg)
qc.h(inp_reg)
qc.x(ancilla)
qc.draw()
```

A.



B.

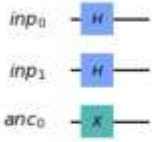


Qiskit

Question 3

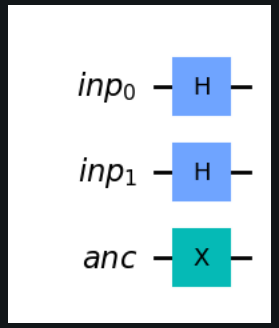
3. Assuming the fragment below, which three code fragments would produce the circuit illustrated?

```
inp_reg = QuantumRegister(2, name='inp')
ancilla = QuantumRegister(1, name='anc')
qc = QuantumCircuit(inp_reg, ancilla)

# Insert code here
```

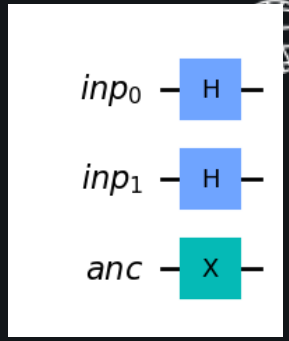


A.
```
qc.h(inp_reg)
qc.x(ancilla)
qc.draw()
```
B.
```
qc.h(inp_reg[0:2])
qc.x(ancilla[0])
qc.draw()
```
C.
```
qc.h(inp_reg[0:1])
qc.x(ancilla[0])
qc.draw()
```
D.
```
qc.h(inp_reg[0])
qc.h(inp_reg[1])
qc.x(ancilla[0])
qc.draw()
```
E.
```
qc.h(inp_reg[1])
qc.h(inp_reg[2])
qc.x(ancilla[1])
qc.draw()
```
F.
```
qc.h(inp_reg)
qc.h(inp_reg)
qc.x(ancilla)
qc.draw()
```

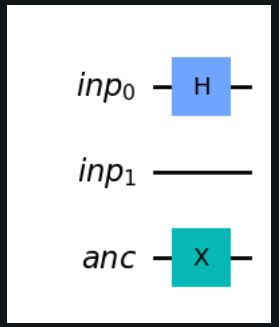
A.



B.



C.

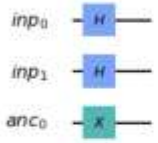


Qiskit

Question 3

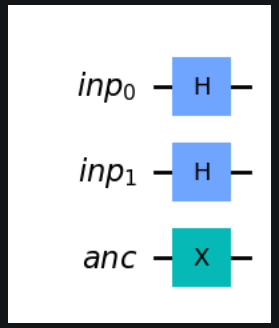
3. Assuming the fragment below, which three code fragments would produce the circuit illustrated?

```
inp_reg = QuantumRegister(2, name='inp')
ancilla = QuantumRegister(1, name='anc')
qc = QuantumCircuit(inp_reg, ancilla)

# Insert code here
```

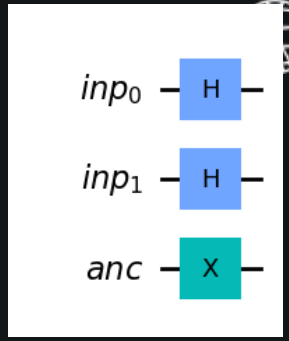


A.
```
qc.h(inp_reg)
qc.x(ancilla)
qc.draw()
```
B.
```
qc.h(inp_reg[0:2])
qc.x(ancilla[0])
qc.draw()
```
C.
```
qc.h(inp_reg[0:1])
qc.x(ancilla[0])
qc.draw()
```
D.
```
qc.h(inp_reg[0])
qc.h(inp_reg[1])
qc.x(ancilla[0])
qc.draw()
```
E.
```
qc.h(inp_reg[1])
qc.h(inp_reg[2])
qc.x(ancilla[1])
qc.draw()
```
F.
```
qc.h(inp_reg)
qc.h(inp_reg)
qc.x(ancilla)
qc.draw()
```

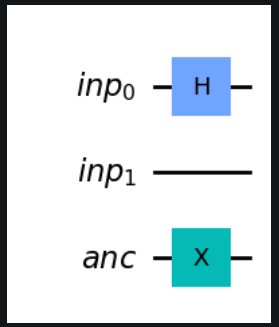
A.



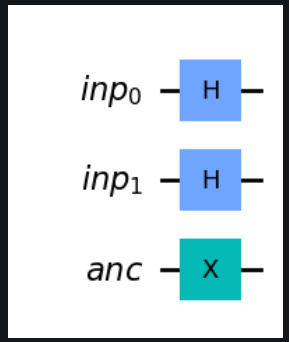
B.



C.



D.

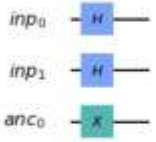


Qiskit

Question 3

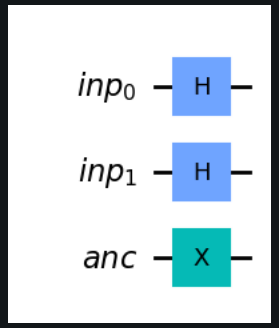
3. Assuming the fragment below, which three code fragments would produce the circuit illustrated?

```
inp_reg = QuantumRegister(2, name='inp')
ancilla = QuantumRegister(1, name='anc')
qc = QuantumCircuit(inp_reg, ancilla)

# Insert code here
```

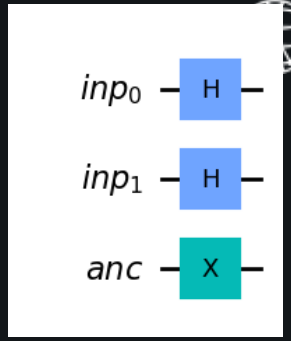


A.
```
qc.h(inp_reg)
qc.x(ancilla)
qc.draw()
```
B.
```
qc.h(inp_reg[0:2])
qc.x(ancilla[0])
qc.draw()
```
C.
```
qc.h(inp_reg[0:1])
qc.x(ancilla[0])
qc.draw()
```
D.
```
qc.h(inp_reg[0])
qc.h(inp_reg[1])
qc.x(ancilla[0])
qc.draw()
```
E.
```
qc.h(inp_reg[1])
qc.h(inp_reg[2])
qc.x(ancilla[1])
qc.draw()
```
F.
```
qc.h(inp_reg)
qc.h(inp_reg)
qc.x(ancilla)
qc.draw()
```

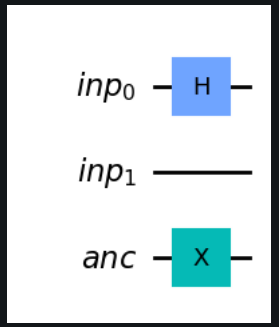
A.



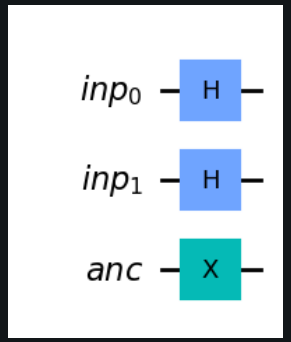
B.



C.



D.



E.

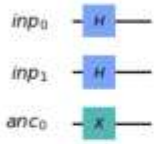
```
Error: list index out of range
```

Qiskit

Question 3

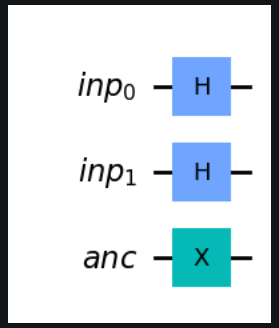
3. Assuming the fragment below, which three code fragments would produce the circuit illustrated?

```
inp_reg = QuantumRegister(2, name='inp')
ancilla = QuantumRegister(1, name='anc')
qc = QuantumCircuit(inp_reg, ancilla)

# Insert code here
```

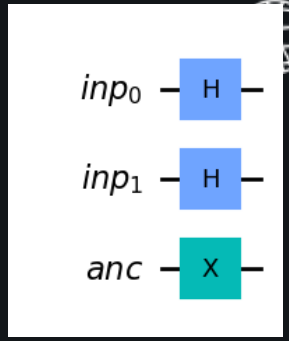


A.
```
qc.h(inp_reg)
qc.x(ancilla)
qc.draw()
```
B.
```
qc.h(inp_reg[0:2])
qc.x(ancilla[0])
qc.draw()
```
C.
```
qc.h(inp_reg[0:1])
qc.x(ancilla[0])
qc.draw()
```
D.
```
qc.h(inp_reg[0])
qc.h(inp_reg[1])
qc.x(ancilla[0])
qc.draw()
```
E.
```
qc.h(inp_reg[1])
qc.h(inp_reg[2])
qc.x(ancilla[1])
qc.draw()
```
F.
```
qc.h(inp_reg)
qc.h(inp_reg)
qc.x(ancilla)
qc.draw()
```

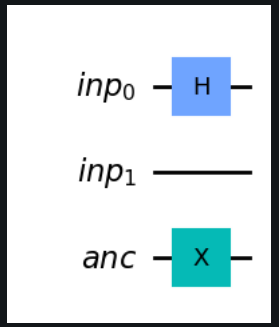
A.



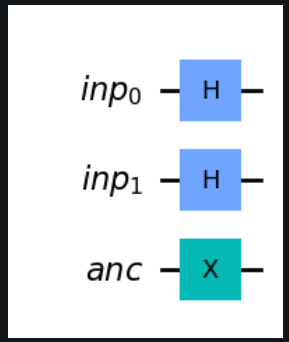
B.



C.



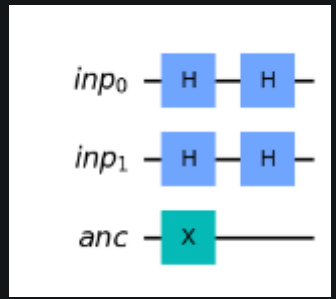
D.



E.

```
Error: list index out of range
```

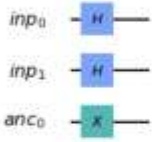
F.



Question 3

3. Assuming the fragment below, which three code fragments would produce the circuit illustrated?

```
inp_reg = QuantumRegister(2, name='inp')
ancilla = QuantumRegister(1, name='anc')
qc = QuantumCircuit(inp_reg, ancilla)

# Insert code here
```



A.
```
qc.h(inp_reg)
qc.x(ancilla)
qc.draw()
```

B.
```
qc.h(inp_reg[0:2])
qc.x(ancilla[0])
qc.draw()
```

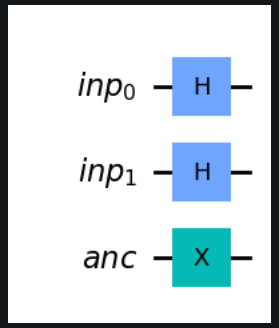
C.
```
qc.h(inp_reg[0:1])
qc.x(ancilla[0])
qc.draw()
```

D.
```
qc.h(inp_reg[0])
qc.h(inp_reg[1])
qc.x(ancilla[0])
qc.draw()
```

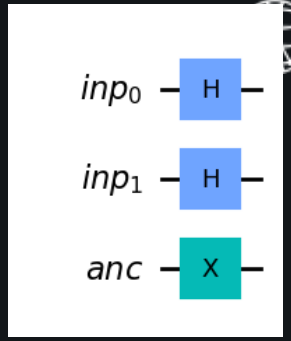
E.
```
qc.h(inp_reg[1])
qc.h(inp_reg[2])
qc.x(ancilla[1])
qc.draw()
```

F.
```
qc.h(inp_reg)
qc.h(inp_reg)
qc.x(ancilla)
qc.draw()
```

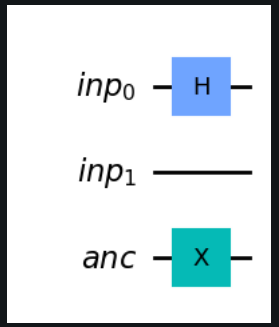
A.



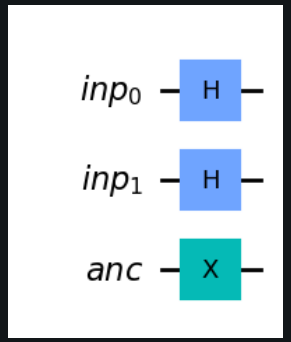
B.



C.



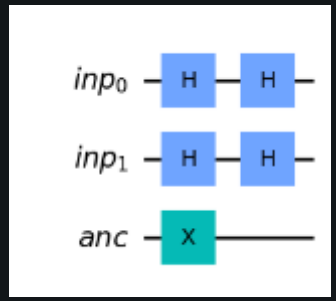
D.



E.

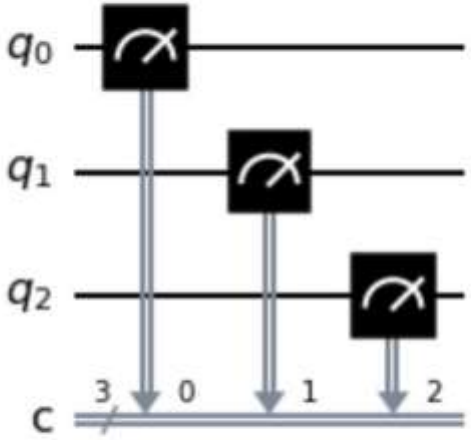
```
Error: list index out of range
```

F.



Question 4

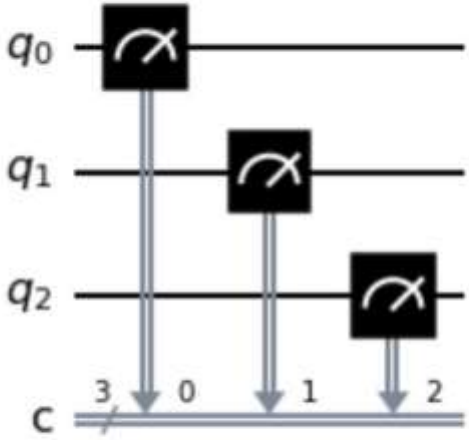
4. Given an empty QuantumCircuit object, qc, with three qubits and three classical bits, which one of these code fragments would create this circuit?



A. `qc.measure([0,1,2], [0,1,2])`
B. `qc.measure([0,0], [1,1], [2,2])`
C. `qc.measure_all()`
D. `qc.measure(0,1,2)`

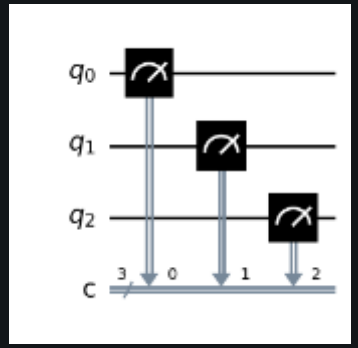
Question 4

4. Given an empty QuantumCircuit object, qc, with three qubits and three classical bits, which one of these code fragments would create this circuit?



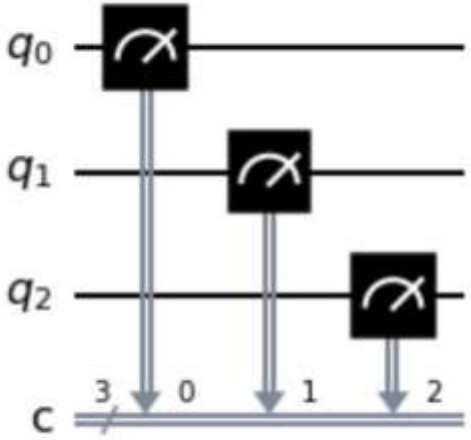
A. `qc.measure([0,1,2], [0,1,2])`
B. `qc.measure([0,0], [1,1], [2,2])`
C. `qc.measure_all()`
D. `qc.measure(0,1,2)`

A.



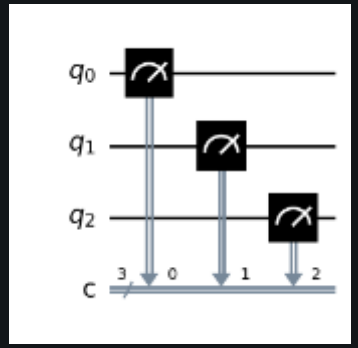
Question 4

4. Given an empty QuantumCircuit object, qc, with three qubits and three classical bits, which one of these code fragments would create this circuit?



A. `qc.measure([0,1,2], [0,1,2])`
B. `qc.measure([0,0], [1,1], [2,2])`
C. `qc.measure_all()`
D. `qc.measure(0,1,2)`

A.

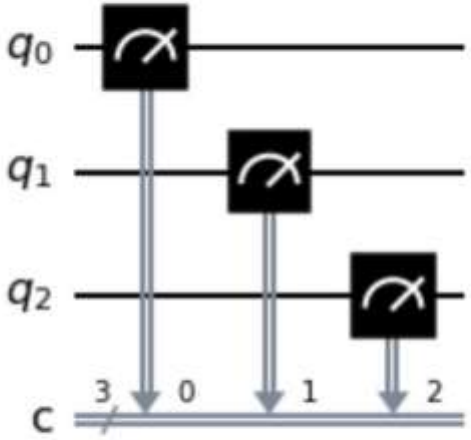


B.

```
Error: QuantumCircuit.measure() takes 3 positional arguments but 4 were given
```

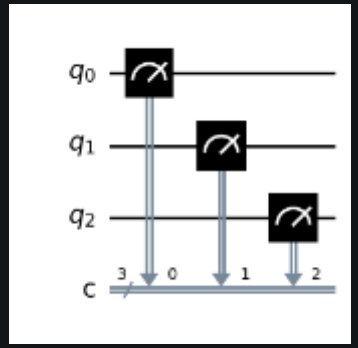
Question 4

4. Given an empty QuantumCircuit object, qc, with three qubits and three classical bits, which one of these code fragments would create this circuit?



A. `qc.measure([0,1,2], [0,1,2])`
B. `qc.measure([0,0], [1,1], [2,2])`
C. `qc.measure_all()`
D. `qc.measure(0,1,2)`

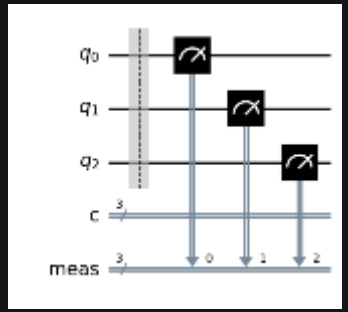
A.



B.

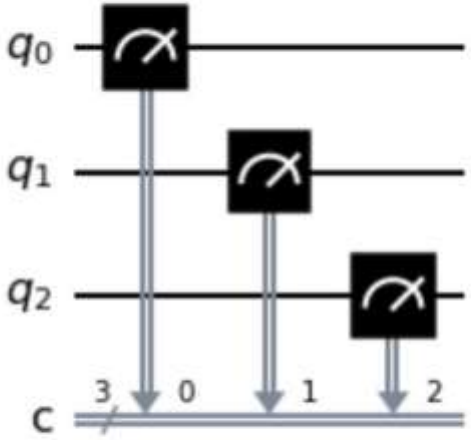
```
Error: QuantumCircuit.measure() takes 3 positional arguments but 4 were given
```

C.



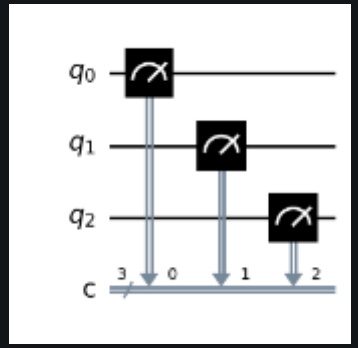
Question 4

4. Given an empty QuantumCircuit object, qc, with three qubits and three classical bits, which one of these code fragments would create this circuit?



A. `qc.measure([0,1,2], [0,1,2])`
B. `qc.measure([0,0], [1,1], [2,2])`
C. `qc.measure_all()`
D. `qc.measure(0,1,2)`

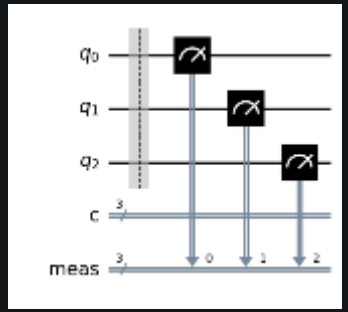
A.



B.

```
Error: QuantumCircuit.measure() takes 3 positional arguments but 4 were given
```

C.

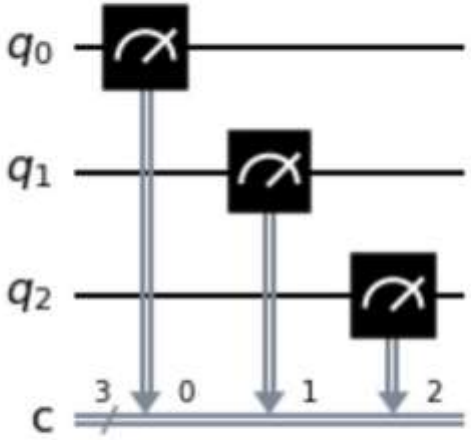


D.

```
Error: QuantumCircuit.measure() takes 3 positional arguments but 4 were given
```

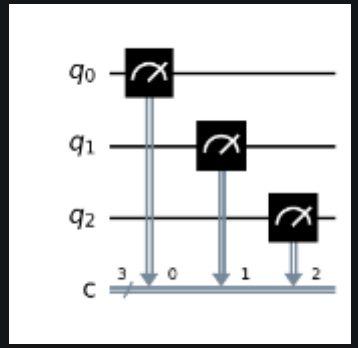
Question 4

4. Given an empty QuantumCircuit object, qc, with three qubits and three classical bits, which one of these code fragments would create this circuit?



A. `qc.measure([0,1,2], [0,1,2])`
B. `qc.measure([0,0], [1,1], [2,2])`
C. `qc.measure_all()`
D. `qc.measure(0,1,2)`

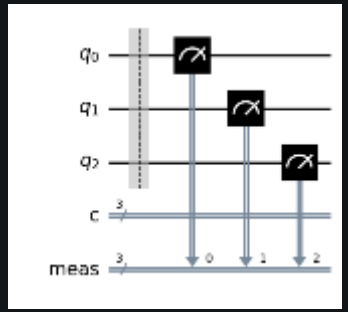
A.



B.

```
Error: QuantumCircuit.measure() takes 3 positional arguments but 4 were given
```

C.



D.

```
Error: QuantumCircuit.measure() takes 3 positional arguments but 4 were given
```

Question 5

Qiskit

5. Which code fragment will produce a maximally entangled, or Bell, state?

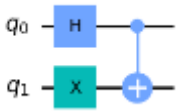
A.
```
bell = QuantumCircuit(2)
bell.h(0)
bell.x(1)
bell.cx(0, 1)
```
B.
```
bell = QuantumCircuit(2)
bell.cx(0, 1)
bell.h(0)
bell.x(1)
```
C.
```
bell = QuantumCircuit(2)
bell.h(0)
bell.x(1)
bell.cz(0, 1)
```
D.
```
bell = QuantumCircuit(2)
bell.h(0)
bell.h(0)
```

Question 5

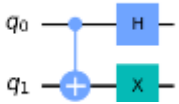
- $|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$
- $|\Phi^-\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle)$
- $|\Psi^+\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)$
- $|\Psi^-\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle)$

5. Which code fragment will produce a maximally entangled, or Bell, state?

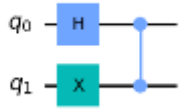
A.
```
bell = QuantumCircuit(2)
bell.h(0)
bell.x(1)
bell.cx(0, 1)
```



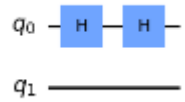
B.
```
bell = QuantumCircuit(2)
bell.cx(0, 1)
bell.h(0)
bell.x(1)
```



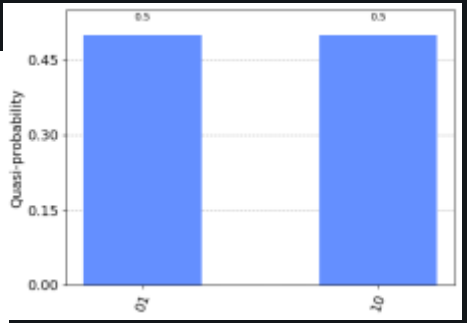
C.
```
bell = QuantumCircuit(2)
bell.h(0)
bell.x(1)
bell.cz(0, 1)
```



D.
```
bell = QuantumCircuit(2)
bell.h(0)
bell.h(0)
```



A.

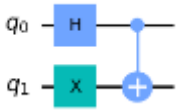


Question 5

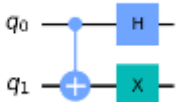
- $|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$
- $|\Phi^-\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle)$
- $|\Psi^+\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)$
- $|\Psi^-\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle)$

5. Which code fragment will produce a maximally entangled, or Bell, state?

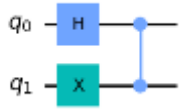
A.
```
bell = QuantumCircuit(2)
bell.h(0)
bell.x(1)
bell.cx(0, 1)
```



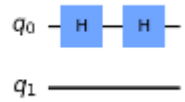
B.
```
bell = QuantumCircuit(2)
bell.cx(0, 1)
bell.h(0)
bell.x(1)
```



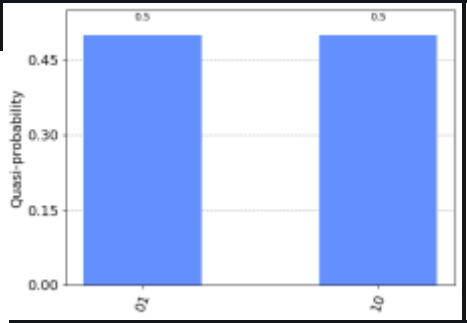
C.
```
bell = QuantumCircuit(2)
bell.h(0)
bell.x(1)
bell.cz(0, 1)
```



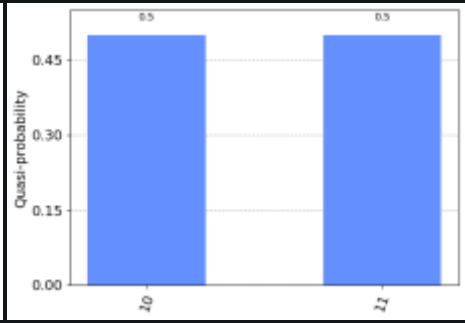
D.
```
bell = QuantumCircuit(2)
bell.h(0)
bell.h(0)
```



A.



B.

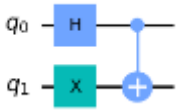


Question 5

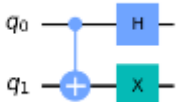
- $|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$
- $|\Phi^-\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle)$
- $|\Psi^+\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)$
- $|\Psi^-\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle)$

5. Which code fragment will produce a maximally entangled, or Bell, state?

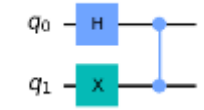
A.
```
bell = QuantumCircuit(2)
bell.h(0)
bell.x(1)
bell.cx(0, 1)
```



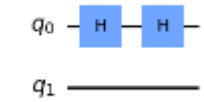
B.
```
bell = QuantumCircuit(2)
bell.cx(0, 1)
bell.h(0)
bell.x(1)
```



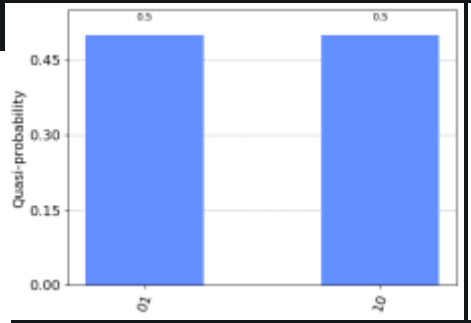
C.
```
bell = QuantumCircuit(2)
bell.h(0)
bell.x(1)
bell.cz(0, 1)
```



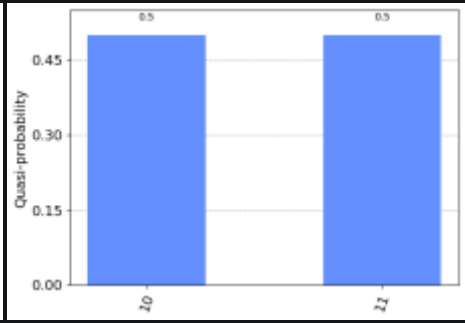
D.
```
bell = QuantumCircuit(2)
bell.h(0)
bell.h(0)
```



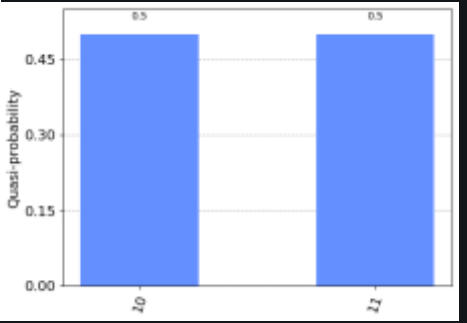
A.



B.



C.

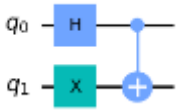


Question 5

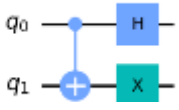
- $|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$
- $|\Phi^-\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle)$
- $|\Psi^+\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)$
- $|\Psi^-\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle)$

5. Which code fragment will produce a maximally entangled, or Bell, state?

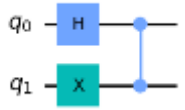
A.
```
bell = QuantumCircuit(2)
bell.h(0)
bell.x(1)
bell.cx(0, 1)
```



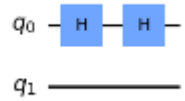
B.
```
bell = QuantumCircuit(2)
bell.cx(0, 1)
bell.h(0)
bell.x(1)
```



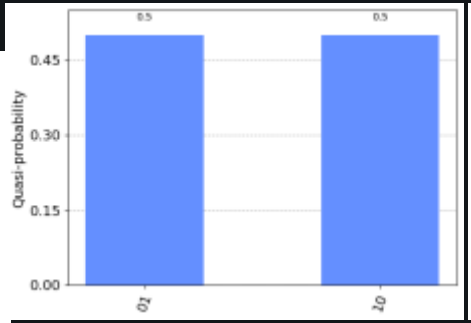
C.
```
bell = QuantumCircuit(2)
bell.h(0)
bell.x(1)
bell.cz(0, 1)
```



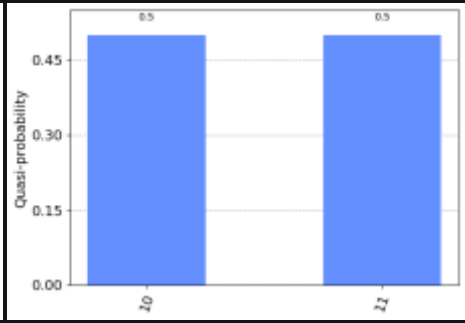
D.
```
bell = QuantumCircuit(2)
bell.h(0)
bell.h(0)
```



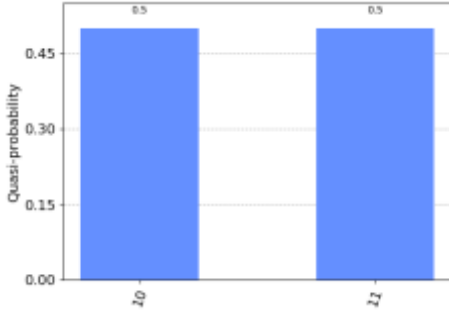
A.



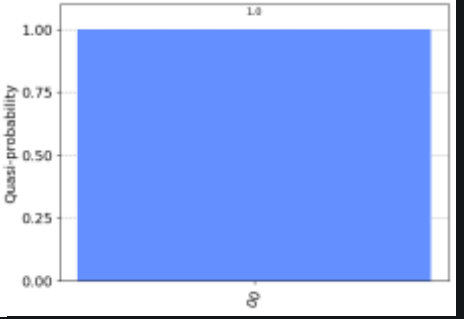
B.



C.



D.

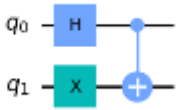


Question 5

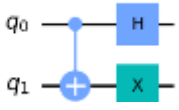
- $|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$
- $|\Phi^-\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle)$
- $|\Psi^+\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)$
- $|\Psi^-\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle)$

5. Which code fragment will produce a maximally entangled, or Bell, state?

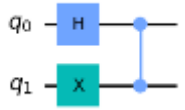
A.
```
bell = QuantumCircuit(2)
bell.h(0)
bell.x(1)
bell.cx(0, 1)
```



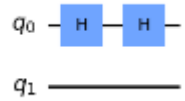
B.
```
bell = QuantumCircuit(2)
bell.cx(0, 1)
bell.h(0)
bell.x(1)
```



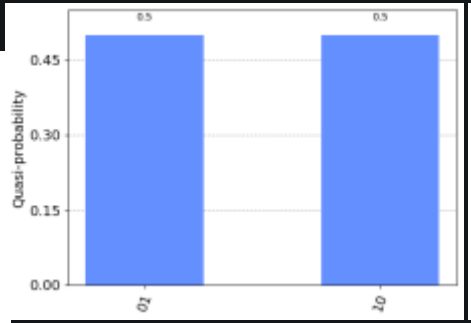
C.
```
bell = QuantumCircuit(2)
bell.h(0)
bell.x(1)
bell.cz(0, 1)
```



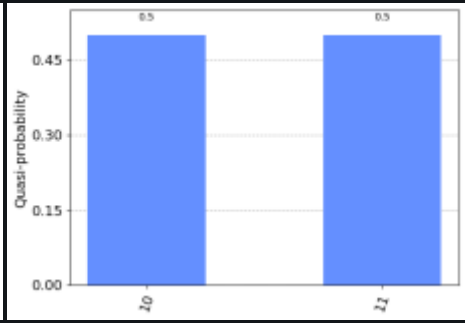
D.
```
bell = QuantumCircuit(2)
bell.h(0)
bell.h(0)
```



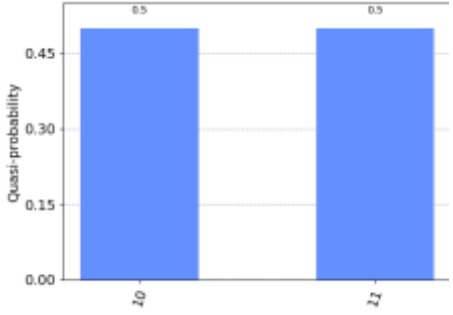
A.



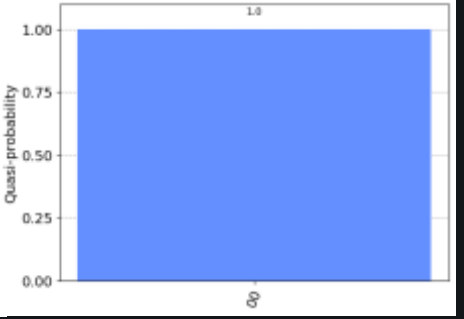
B.



C.



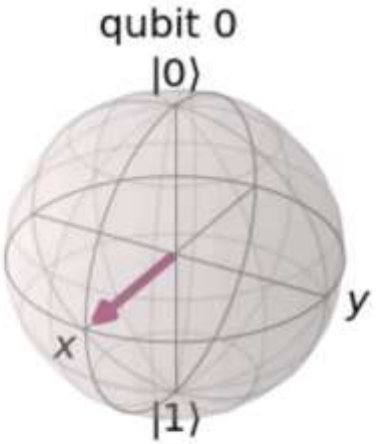
D.



Question 6

6. Given this code, which two inserted code fragments result in the state vector represented by this Bloch sphere?

```
qc = QuantumCircuit(1,1)
# Insert code fragment here

simulator = Aer.get_backend('statevector_simulator')
job = execute(qc, simulator)
result = job.result()
outputstate = result.get_statevector(qc)
plot_bloch_multivector(outputstate)
```

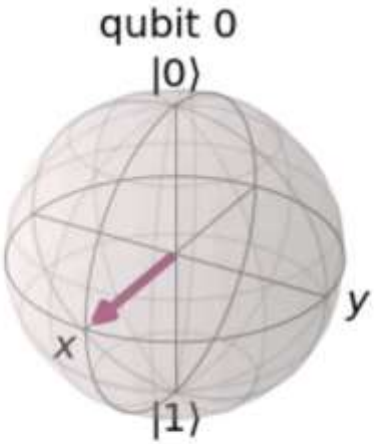


A. `qc.h(0)`
B. `qc.rx(math.pi / 2, 0)`
C. `qc.ry(math.pi / 2, 0)`
D. `qc.rx(math.pi / 2, 0)`
`qc.rz(-math.pi / 2, 0)`
E. `qc.ry(math.pi, 0)`

Question 6

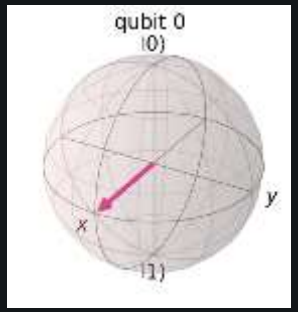
6. Given this code, which two inserted code fragments result in the state vector represented by this Bloch sphere?

```
qc = QuantumCircuit(1,1)
# Insert code fragment here

simulator = Aer.get_backend('statevector_simulator')
job = execute(qc, simulator)
result = job.result()
outputstate = result.get_statevector(qc)
plot_bloch_multivector(outputstate)
```



A. `qc.h(0)`
B. `qc.rx(math.pi / 2, 0)`
C. `qc.ry(math.pi / 2, 0)`
D. `qc.rx(math.pi / 2, 0)`
`qc.rz(-math.pi / 2, 0)`
E. `qc.ry(math.pi, 0)`

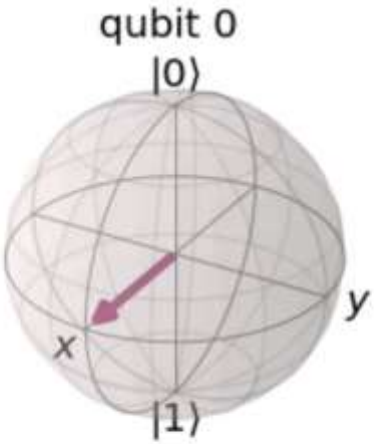
A.



Question 6

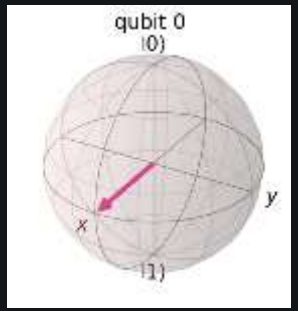
6. Given this code, which two inserted code fragments result in the state vector represented by this Bloch sphere?

```
qc = QuantumCircuit(1,1)
# Insert code fragment here

simulator = Aer.get_backend('statevector_simulator')
job = execute(qc, simulator)
result = job.result()
outputstate = result.get_statevector(qc)
plot_bloch_multivector(outputstate)
```

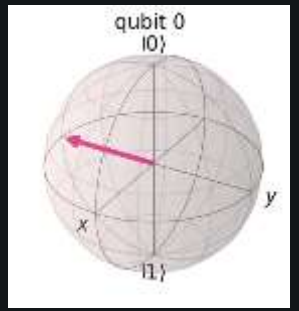


A. `qc.h(0)`
B. `qc.rx(math.pi / 2, 0)`
C. `qc.ry(math.pi / 2, 0)`
D. `qc.rx(math.pi / 2, 0)`
`qc.rz(-math.pi / 2, 0)`
E. `qc.ry(math.pi, 0)`

A.



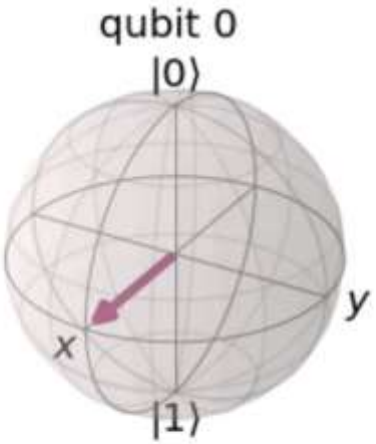
B.



Question 6

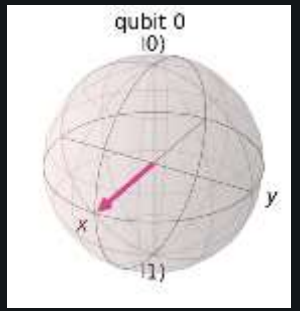
6. Given this code, which two inserted code fragments result in the state vector represented by this Bloch sphere?

```
qc = QuantumCircuit(1,1)
# Insert code fragment here

simulator = Aer.get_backend('statevector_simulator')
job = execute(qc, simulator)
result = job.result()
outputstate = result.get_statevector(qc)
plot_bloch_multivector(outputstate)
```

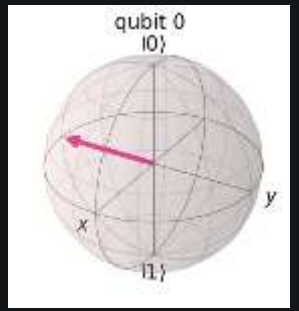


A. `qc.h(0)`
B. `qc.rx(math.pi / 2, 0)`
C. `qc.ry(math.pi / 2, 0)`
D. `qc.rx(math.pi / 2, 0)`
`qc.rz(-math.pi / 2, 0)`
E. `qc.ry(math.pi, 0)`

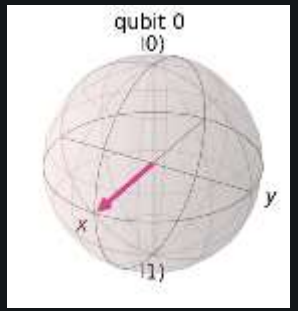
A.



B.



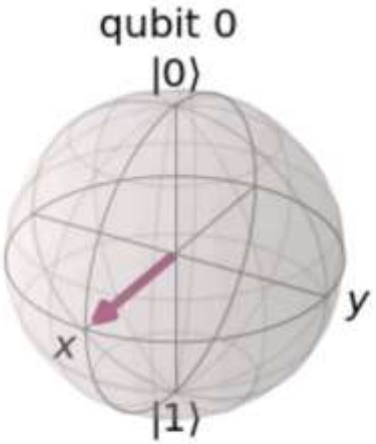
C.



Question 6

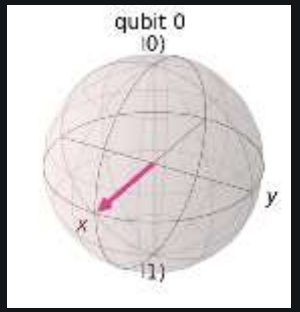
6. Given this code, which two inserted code fragments result in the state vector represented by this Bloch sphere?

```
qc = QuantumCircuit(1,1)
# Insert code fragment here

simulator = Aer.get_backend('statevector_simulator')
job = execute(qc, simulator)
result = job.result()
outputstate = result.get_statevector(qc)
plot_bloch_multivector(outputstate)
```

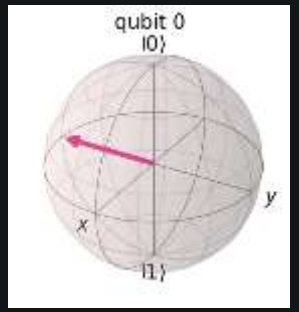


A. `qc.h(0)`
B. `qc.rx(math.pi / 2, 0)`
C. `qc.ry(math.pi / 2, 0)`
D. `qc.rx(math.pi / 2, 0)`
`qc.rz(-math.pi / 2, 0)`
E. `qc.ry(math.pi, 0)`

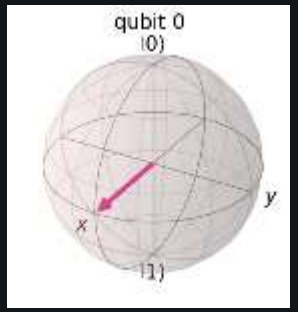
A.



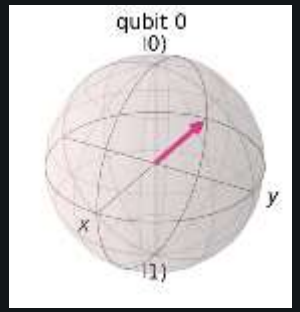
B.



C.



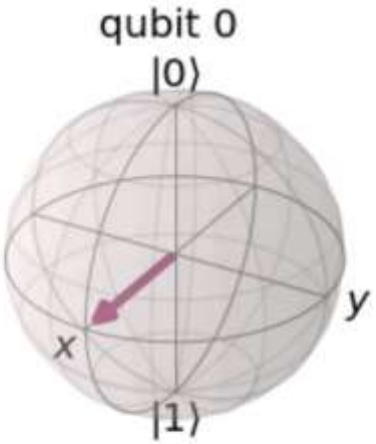
D.



Question 6

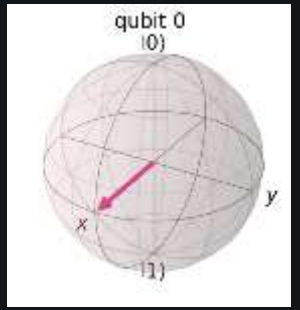
6. Given this code, which two inserted code fragments result in the state vector represented by this Bloch sphere?

```
qc = QuantumCircuit(1,1)
# Insert code fragment here

simulator = Aer.get_backend('statevector_simulator')
job = execute(qc, simulator)
result = job.result()
outputstate = result.get_statevector(qc)
plot_bloch_multivector(outputstate)
```

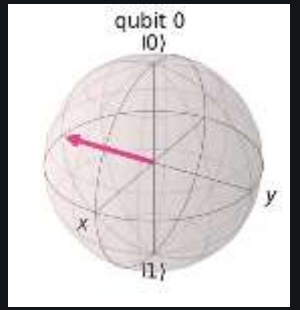


A. `qc.h(0)`
B. `qc.rx(math.pi / 2, 0)`
C. `qc.ry(math.pi / 2, 0)`
D. `qc.rx(math.pi / 2, 0)`
`qc.rz(-math.pi / 2, 0)`
E. `qc.ry(math.pi, 0)`

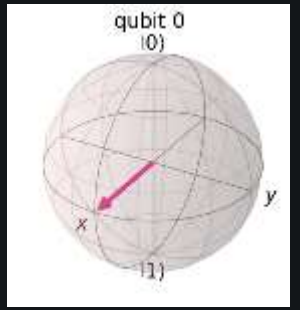
A.



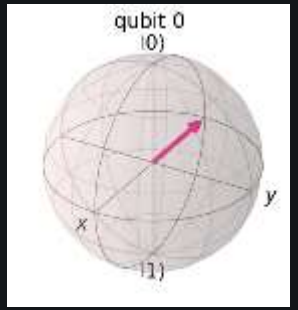
B.



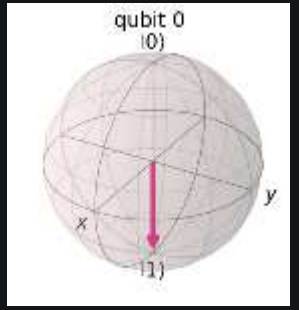
C.



D.



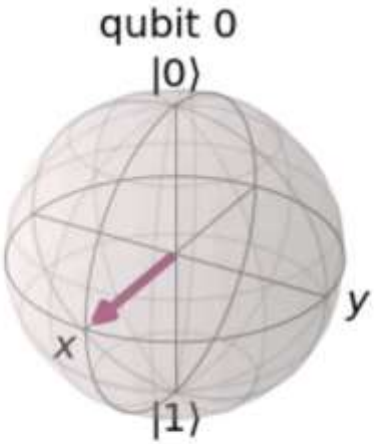
E.



Question 6

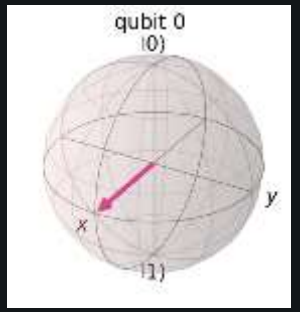
6. Given this code, which two inserted code fragments result in the state vector represented by this Bloch sphere?

```
qc = QuantumCircuit(1,1)
# Insert code fragment here

simulator = Aer.get_backend('statevector_simulator')
job = execute(qc, simulator)
result = job.result()
outputstate = result.get_statevector(qc)
plot_bloch_multivector(outputstate)
```

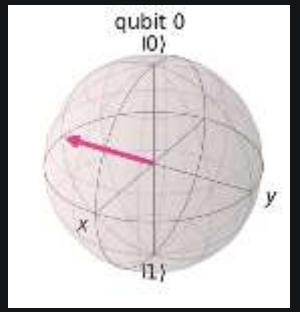


A. `qc.h(0)`
B. `qc.rx(math.pi / 2, 0)`
C. `qc.ry(math.pi / 2, 0)`
D. `qc.rx(math.pi / 2, 0)`
`qc.rz(-math.pi / 2, 0)`
E. `qc.ry(math.pi, 0)`

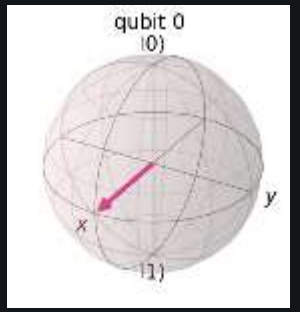
A.



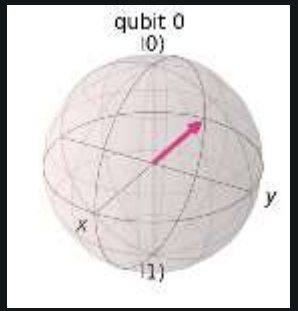
B.



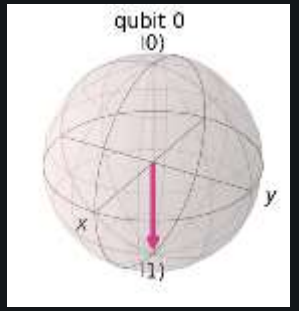
C.



D.



E.



Question 7

Qiskit

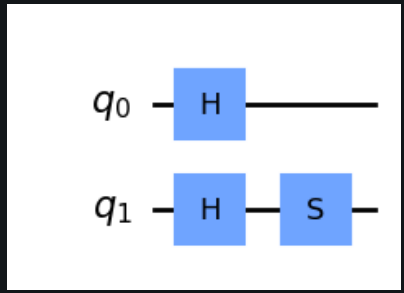
7. **S-gate is a Qiskit phase gate with what value of the phase parameter?**

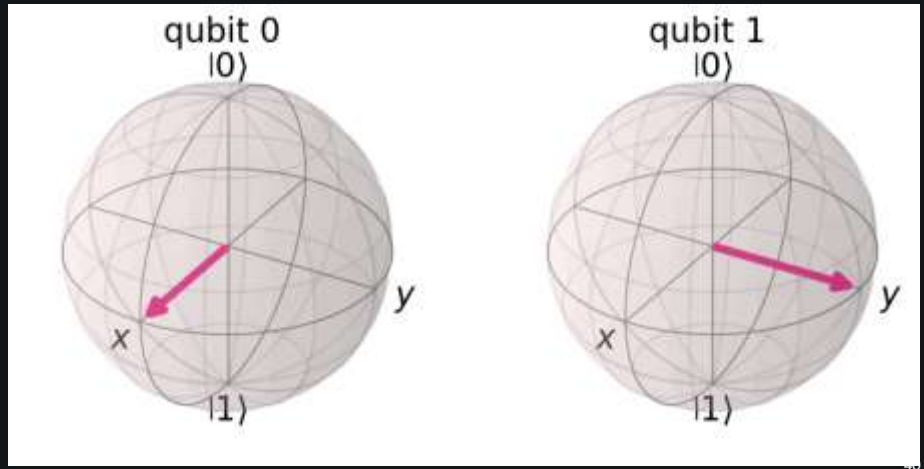
   A. π/4
   B. π/2
   C. π/8
   D. π

Question 7

7. **S-gate is a Qiskit phase gate with what value of the phase parameter?**

   A. π/4
   B. π/2
   C. π/8
   D. π



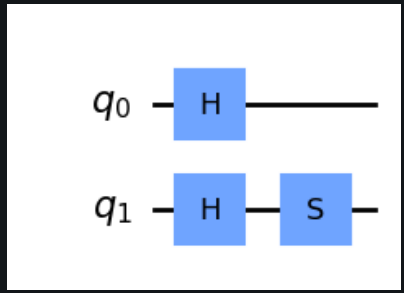


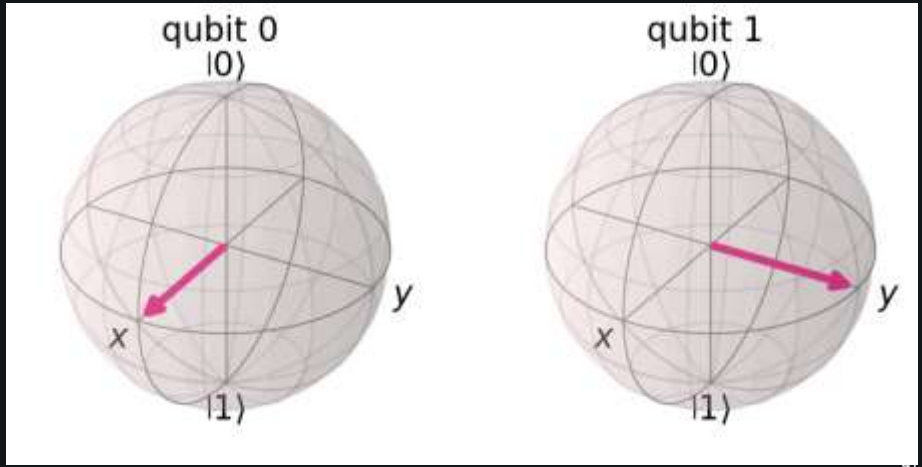
Question 7

Qiskit

7. **S-gate is a Qiskit phase gate with what value of the phase parameter?**

A. π/4
B. π/2
C. π/8
D. π





Question 8

8. Which two code fragments, when inserted into the code below, will produce the statevector shown in the output?

```
from qiskit import QuantumCircuit, Aer, execute
from math import sqrt

qc = QuantumCircuit(2)

# Insert fragment here

simulator = Aer.get_backend('statevector_simulator')
result = execute(qc, simulator).result()
statevector = result.get_statevector()
print(statevector)
```

Output:

```
[0.707+0.j  0.+0.j  0.+0.j  0.707+0.j]
```

A.
```
v = [1/sqrt(2), 0, 0, 1/sqrt(2)]
qc.initialize(v,[0,1])
```

B.
```
qc.h(0)
qc.cx(0,1)
```

C.
```
v1, v2 = [1,0], [0,1]
qc.initialize(v1,0)
qc.initialize(v2,1)
```

D.
```
qc.cx(0,1)
qc.measure_all()
```

E.
```
qc.h(0)
qc.h(1)
qc.measure_all()
```

Question 8

Qiskit

8. Which two code fragments, when inserted into the code below, will produce the statevector shown in the output?

```
from qiskit import QuantumCircuit, Aer, execute
from math import sqrt

qc = QuantumCircuit(2)

# Insert fragment here

simulator = Aer.get_backend('statevector_simulator')
result = execute(qc, simulator).result()
statevector = result.get_statevector()
print(statevector)
```

Output:

$$\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

```
[0.707+0.j  0.+0.j  0.+0.j  0.707+0.j]
```

A.
```
v = [1/sqrt(2), 0, 0, 1/sqrt(2)]
qc.initialize(v,[0,1])
```
B.
```
qc.h(0)
qc.cx(0,1)
```
C.
```
v1, v2 = [1,0], [0,1]
qc.initialize(v1,0)
qc.initialize(v2,1)
```
D.
```
qc.cx(0,1)
qc.measure_all()
```
E.
```
qc.h(0)
qc.h(1)
qc.measure_all()
```

A.

```
Statevector([0.70710678+0.j, 0.        +0.j, 0.        +0.j,
             0.70710678+0.j],
            dims=(2, 2))
```

Question 8

Qiskit

8. Which two code fragments, when inserted into the code below, will produce the statevector shown in the output?

```
from qiskit import QuantumCircuit, Aer, execute
from math import sqrt

qc = QuantumCircuit(2)

# Insert fragment here

simulator = Aer.get_backend('statevector_simulator')
result = execute(qc, simulator).result()
statevector = result.get_statevector()
print(statevector)
```

Output: $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$

```
[0.707+0.j  0.+0.j  0.+0.j  0.707+0.j]
```

A.
```
v = [1/sqrt(2), 0, 0, 1/sqrt(2)]
qc.initialize(v,[0,1])
```

B.
```
qc.h(0)
qc.cx(0,1)
```

C.
```
v1, v2 = [1,0], [0,1]
qc.initialize(v1,0)
qc.initialize(v2,1)
```

D.
```
qc.cx(0,1)
qc.measure_all()
```

E.
```
qc.h(0)
qc.h(1)
qc.measure_all()
```

A.

```
Statevector([0.70710678+0.j, 0.        +0.j, 0.        +0.j,
             0.70710678+0.j],
            dims=(2, 2))
```

B.

```
Statevector([0.70710678+0.j, 0.        +0.j, 0.        +0.j,
             0.70710678+0.j],
            dims=(2, 2))
```

Question 8

Qiskit

8. Which two code fragments, when inserted into the code below, will produce the statevector shown in the output?

```
from qiskit import QuantumCircuit, Aer, execute
from math import sqrt

qc = QuantumCircuit(2)

# Insert fragment here

simulator = Aer.get_backend('statevector_simulator')
result = execute(qc, simulator).result()
statevector = result.get_statevector()
print(statevector)
```

Output: $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$

```
[0.707+0.j  0.+0.j  0.+0.j  0.707+0.j]
```

A.
```
v = [1/sqrt(2), 0, 0, 1/sqrt(2)]
qc.initialize(v,[0,1])
```

B.
```
qc.h(0)
qc.cx(0,1)
```

C.
```
v1, v2 = [1,0], [0,1]
qc.initialize(v1,0)
qc.initialize(v2,1)
```

D.
```
qc.cx(0,1)
qc.measure_all()
```

E.
```
qc.h(0)
qc.h(1)
qc.measure_all()
```

A.

```
Statevector([0.70710678+0.j, 0.        +0.j, 0.        +0.j,
             0.70710678+0.j],
            dims=(2, 2))
```

B.

```
Statevector([0.70710678+0.j, 0.        +0.j, 0.        +0.j,
             0.70710678+0.j],
            dims=(2, 2))
```

C.

```
Statevector([0.+0.j, 0.+0.j, 1.+0.j, 0.+0.j],
            dims=(2, 2))
```

Question 8

8. Which two code fragments, when inserted into the code below, will produce the statevector shown in the output?

```
from qiskit import QuantumCircuit, Aer, execute
from math import sqrt

qc = QuantumCircuit(2)

# Insert fragment here

simulator = Aer.get_backend('statevector_simulator')
result = execute(qc, simulator).result()
statevector = result.get_statevector()
print(statevector)
```

Output:

$$\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

```
[0.707+0.j  0.+0.j  0.+0.j  0.707+0.j]
```

A.
```
v = [1/sqrt(2), 0, 0, 1/sqrt(2)]
qc.initialize(v,[0,1])
```
B.
```
qc.h(0)
qc.cx(0,1)
```
C.
```
v1, v2 = [1,0], [0,1]
qc.initialize(v1,0)
qc.initialize(v2,1)
```
D.
```
qc.cx(0,1)
qc.measure_all()
```
E.
```
qc.h(0)
qc.h(1)
qc.measure_all()
```

A.

```
Statevector([0.70710678+0.j, 0.        +0.j, 0.        +0.j,
             0.70710678+0.j],
            dims=(2, 2))
```

B.

```
Statevector([0.70710678+0.j, 0.        +0.j, 0.        +0.j,
             0.70710678+0.j],
            dims=(2, 2))
```

C.

```
Statevector([0.+0.j, 0.+0.j, 1.+0.j, 0.+0.j],
            dims=(2, 2))
```

D.

```
Statevector([1.+0.j, 0.+0.j, 0.+0.j, 0.+0.j],
            dims=(2, 2))
```

Question 8

8. Which two code fragments, when inserted into the code below, will produce the statevector shown in the output?

```
from qiskit import QuantumCircuit, Aer, execute
from math import sqrt

qc = QuantumCircuit(2)

# Insert fragment here

simulator = Aer.get_backend('statevector_simulator')
result = execute(qc, simulator).result()
statevector = result.get_statevector()
print(statevector)
```

Output:

$$\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

```
[0.707+0.j  0.+0.j  0.+0.j  0.707+0.j]
```

A.
```
v = [1/sqrt(2), 0, 0, 1/sqrt(2)]
qc.initialize(v,[0,1])
```
B.
```
qc.h(0)
qc.cx(0,1)
```
C.
```
v1, v2 = [1,0], [0,1]
qc.initialize(v1,0)
qc.initialize(v2,1)
```
D.
```
qc.cx(0,1)
qc.measure_all()
```
E.
```
qc.h(0)
qc.h(1)
qc.measure_all()
```

A.

```
Statevector([0.70710678+0.j, 0.        +0.j, 0.        +0.j,
             0.70710678+0.j],
            dims=(2, 2))
```

B.

```
Statevector([0.70710678+0.j, 0.        +0.j, 0.        +0.j,
             0.70710678+0.j],
            dims=(2, 2))
```

C.

```
Statevector([0.+0.j, 0.+0.j, 1.+0.j, 0.+0.j],
            dims=(2, 2))
```

D.

```
Statevector([1.+0.j, 0.+0.j, 0.+0.j, 0.+0.j],
            dims=(2, 2))
```

E.

```
Statevector([0.+0.j, 1.+0.j, 0.+0.j, 0.+0.j],
            dims=(2, 2))
```

Question 8

Qiskit

8. Which two code fragments, when inserted into the code below, will produce the statevector shown in the output?

```
from qiskit import QuantumCircuit, Aer, execute
from math import sqrt

qc = QuantumCircuit(2)

# Insert fragment here

simulator = Aer.get_backend('statevector_simulator')
result = execute(qc, simulator).result()
statevector = result.get_statevector()
print(statevector)
```

Output: $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$

```
[0.707+0.j  0.+0.j  0.+0.j  0.707+0.j]
```

A.
```
v = [1/sqrt(2), 0, 0, 1/sqrt(2)]
qc.initialize(v,[0,1])
```

B.
```
qc.h(0)
qc.cx(0,1)
```

C.
```
v1, v2 = [1,0], [0,1]
qc.initialize(v1,0)
qc.initialize(v2,1)
```

D.
```
qc.cx(0,1)
qc.measure_all()
```

E.
```
qc.h(0)
qc.h(1)
qc.measure_all()
```

A.
```
Statevector([0.70710678+0.j, 0.        +0.j, 0.        +0.j,
             0.70710678+0.j],
            dims=(2, 2))
```

B.
```
Statevector([0.70710678+0.j, 0.        +0.j, 0.        +0.j,
             0.70710678+0.j],
            dims=(2, 2))
```

C.
```
Statevector([0.+0.j, 0.+0.j, 1.+0.j, 0.+0.j],
            dims=(2, 2))
```

D.
```
Statevector([1.+0.j, 0.+0.j, 0.+0.j, 0.+0.j],
            dims=(2, 2))
```

E.
```
Statevector([0.+0.j, 1.+0.j, 0.+0.j, 0.+0.j],
            dims=(2, 2))
```

Question 9

Qiskit

9. Which code fragment will produce a multi-qubit gate other than a CNOT ?

A. `qc.cx(0,1)`
B. `qc.cnot(0,1)`
C. `qc.mct([0],1)`
D. `qc.cz(0,1)`

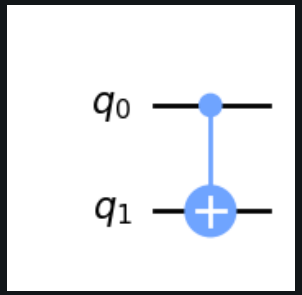
Question 9

Qiskit

9. Which code fragment will produce a multi-qubit gate other than a CNOT ?

A. `qc.cx(0,1)`
B. `qc.cnot(0,1)`
C. `qc.mct([0],1)`
D. `qc.cz(0,1)`

A.



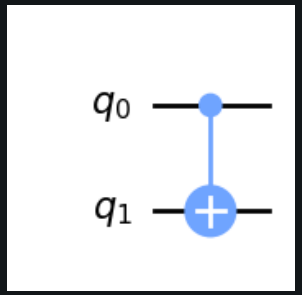
Question 9

Qiskit

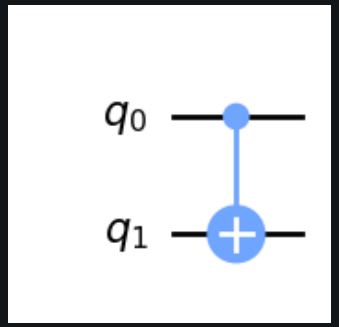
9. Which code fragment will produce a multi-qubit gate other than a CNOT ?

A. `qc.cx(0,1)`
B. `qc.cnot(0,1)`
C. `qc.mct([0],1)`
D. `qc.cz(0,1)`

A.



B.



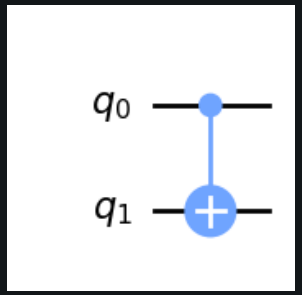
Question 9

Qiskit

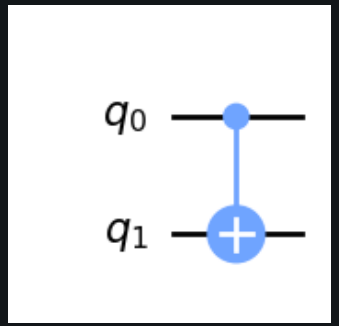
9. Which code fragment will produce a multi-qubit gate other than a CNOT ?

A. `qc.cx(0,1)`
B. `qc.cnot(0,1)`
C. `qc.mct([0],1)`
D. `qc.cz(0,1)`

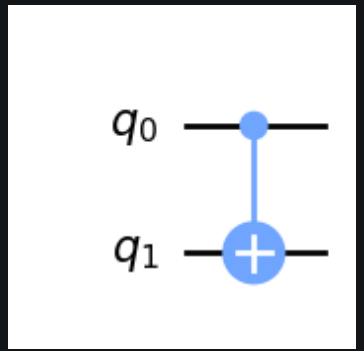
A.



B.



C.



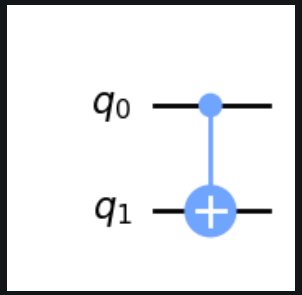
Question 9

Qiskit

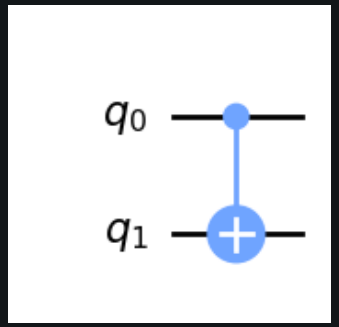
9. Which code fragment will produce a multi-qubit gate other than a CNOT ?

A. `qc.cx(0,1)`
B. `qc.cnot(0,1)`
C. `qc.mct([0],1)`
D. `qc.cz(0,1)`

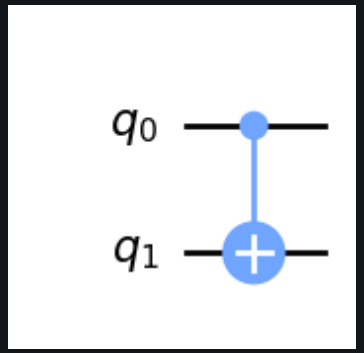
A.



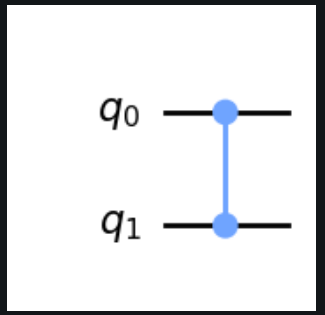
B.



C.



D.



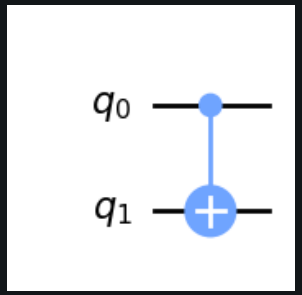
Question 9

Qiskit

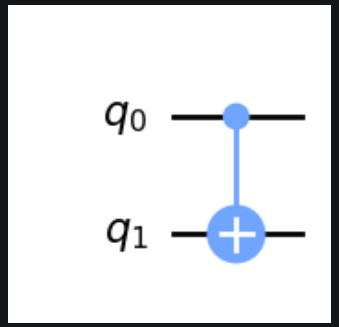
9. Which code fragment will produce a multi-qubit gate other than a CNOT ?

A. `qc.cx(0,1)`
B. `qc.cnot(0,1)`
C. `qc.mct([0],1)`
D. `qc.cz(0,1)`

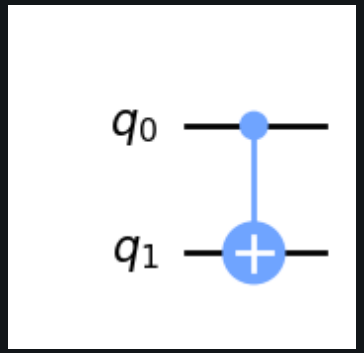
A.



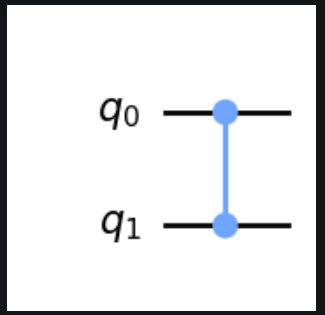
B.



C.



D.



Question 10

10. Which code fragment will produce a multi-qubit gate other than a Toffoli?

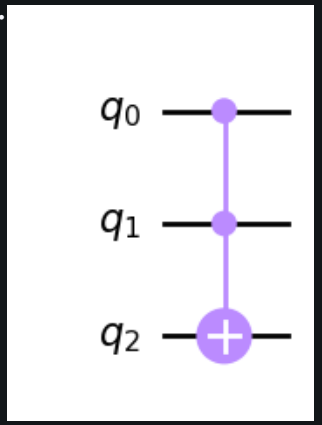
A.
```
qc.ccx(0,1,2)
```
B.
```
qc.mct([0,1], 2)
```
C.
```
from qiskit.circuit.library import CXGate
ccx = CXGate().control()
qc.append(ccx, [0,1,2])
```
D.
```
qc.cry(0,1,2)
```

Question 10

10. Which code fragment will produce a multi-qubit gate other than a Toffoli?

A.
```
qc.ccx(0,1,2)
```
B.
```
qc.mct([0,1], 2)
```
C.
```
from qiskit.circuit.library import CXGate
ccx = CXGate().control()
qc.append(ccx, [0,1,2])
```
D.
```
qc.cry(0,1,2)
```

A.

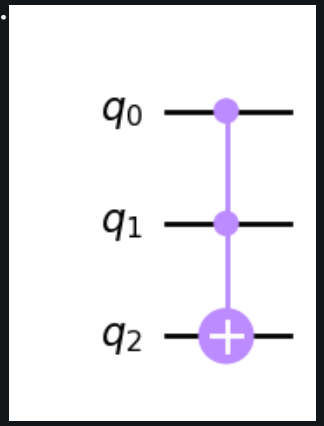


Question 10

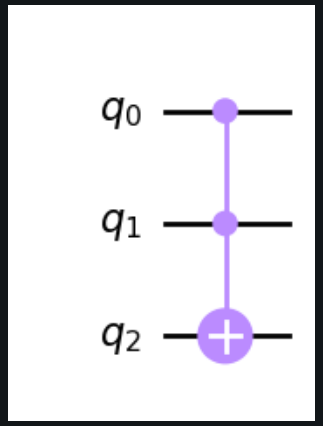
10. Which code fragment will produce a multi-qubit gate other than a Toffoli?

A.
```
qc.ccx(0,1,2)
```
B.
```
qc.mct([0,1], 2)
```
C.
```
from qiskit.circuit.library import CXGate
ccx = CXGate().control()
qc.append(ccx, [0,1,2])
```
D.
```
qc.cry(0,1,2)
```

A.



B.

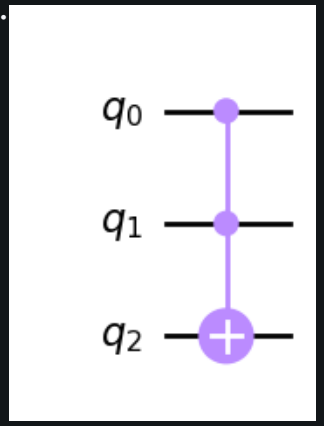


Question 10

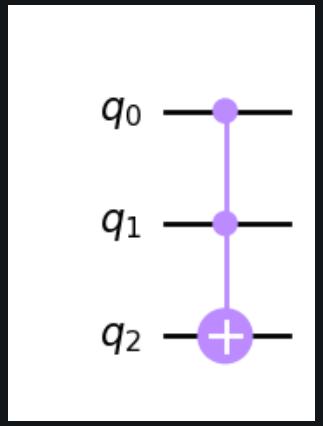
10. Which code fragment will produce a multi-qubit gate other than a Toffoli?

A.
```
qc.ccx(0,1,2)
```
B.
```
qc.mct([0,1], 2)
```
C.
```
from qiskit.circuit.library import CXGate
ccx = CXGate().control()
qc.append(ccx, [0,1,2])
```
D.
```
qc.cry(0,1,2)
```

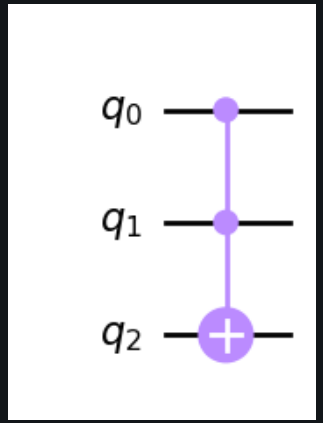
A.



B.



C.

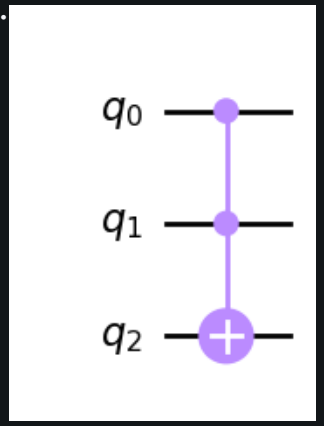


Question 10

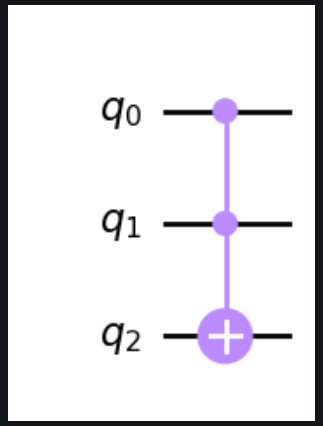
10. Which code fragment will produce a multi-qubit gate other than a Toffoli?

A.
```
qc.ccx(0,1,2)
```
B.
```
qc.mct([0,1], 2)
```
C.
```
from qiskit.circuit.library import CXGate
ccx = CXGate().control()
qc.append(ccx, [0,1,2])
```
D.
```
qc.cry(0,1,2)
```

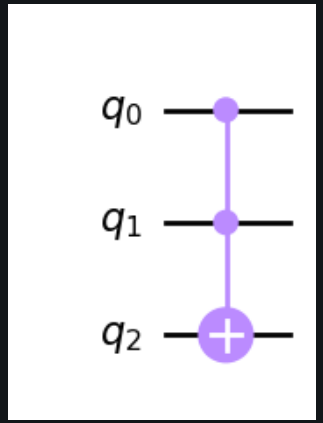
A.



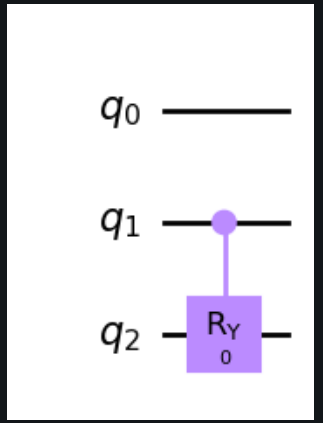
B.



C.



D.

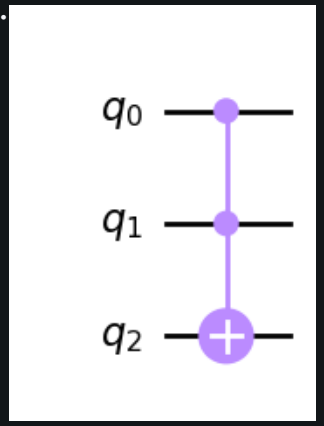


Question 10

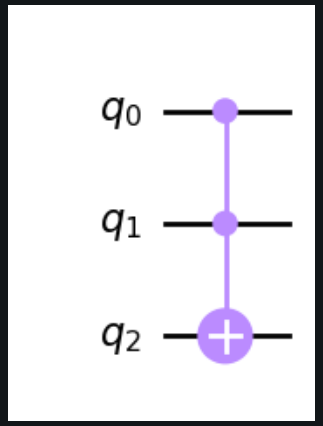
10. Which code fragment will produce a multi-qubit gate other than a Toffoli?

A. `qc.ccx(0,1,2)`
B. `qc.mct([0,1], 2)`
C.
```
from qiskit.circuit.library import CXGate
ccx = CXGate().control()
qc.append(ccx, [0,1,2])
```
D. `qc.cry(0,1,2)`

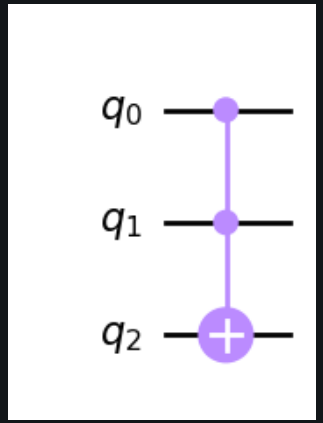
A.



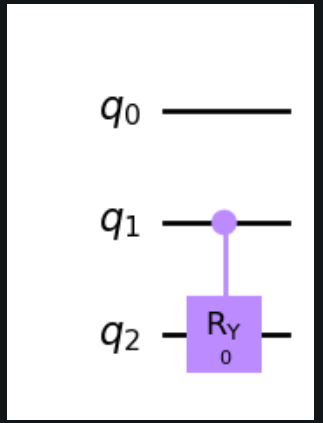
B.



C.



D.



Question 11

Qiskit

11. Which two options would place a barrier across all qubits to the QuantumCircuit below?

```
qc = QuantumCircuit(3,3)
```

A. `qc.barrier(qc)`
B. `qc.barrier([0,1,2])`
C. `qc.barrier()`
D. `qc.barrier(3)`
E. `qc.barrier_all()`

Question 11

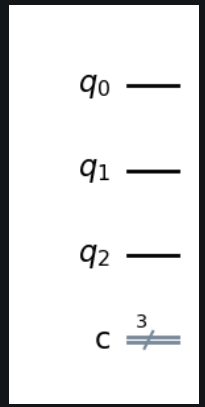
Qiskit

11. Which two options would place a barrier across all qubits to the QuantumCircuit below?

```
qc = QuantumCircuit(3,3)
```

A. `qc.barrier(qc)`
B. `qc.barrier([0,1,2])`
C. `qc.barrier()`
D. `qc.barrier(3)`
E. `qc.barrier_all()`

A.



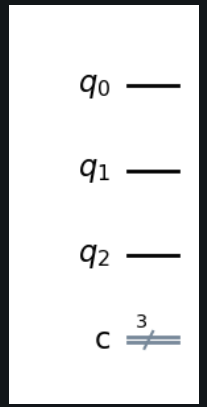
Question 11

11. Which two options would place a barrier across all qubits to the QuantumCircuit below?

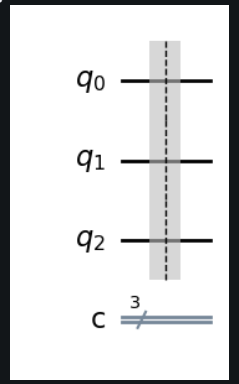
```
qc = QuantumCircuit(3,3)
```

A. `qc.barrier(qc)`
B. `qc.barrier([0,1,2])`
C. `qc.barrier()`
D. `qc.barrier(3)`
E. `qc.barrier_all()`

A.



B.



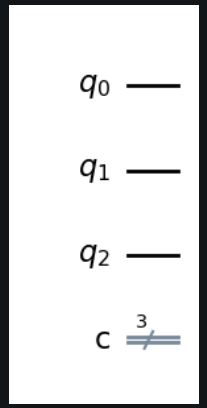
Question 11

11. Which two options would place a barrier across all qubits to the QuantumCircuit below?

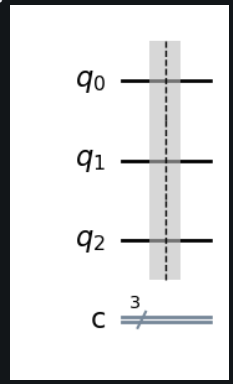
```
qc = QuantumCircuit(3,3)
```

A. `qc.barrier(qc)`
B. `qc.barrier([0,1,2])`
C. `qc.barrier()`
D. `qc.barrier(3)`
E. `qc.barrier_all()`

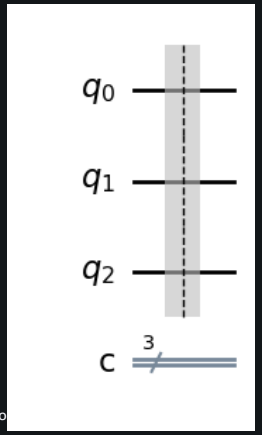
A.



B.



C.



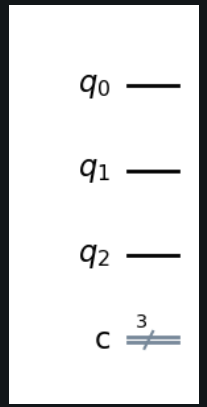
Question 11

11. Which two options would place a barrier across all qubits to the QuantumCircuit below?

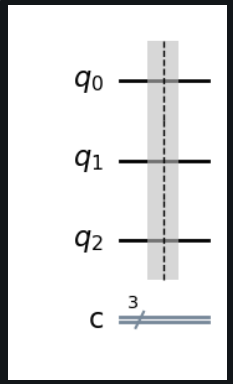
```
qc = QuantumCircuit(3,3)
```

A. `qc.barrier(qc)`
B. `qc.barrier([0,1,2])`
C. `qc.barrier()`
D. `qc.barrier(3)`
E. `qc.barrier_all()`

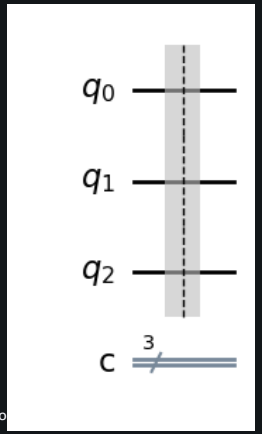
A.



B.



C.



D.

```
This solution raised the following error: 'Index 3 out of range for size 3.'
```

Question 11

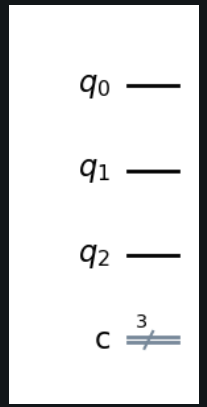
Qiskit

11. Which two options would place a barrier across all qubits to the QuantumCircuit below?

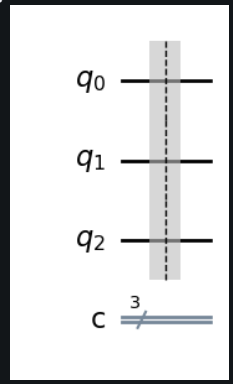
```
qc = QuantumCircuit(3,3)
```

A. `qc.barrier(qc)`
B. `qc.barrier([0,1,2])`
C. `qc.barrier()`
D. `qc.barrier(3)`
E. `qc.barrier_all()`

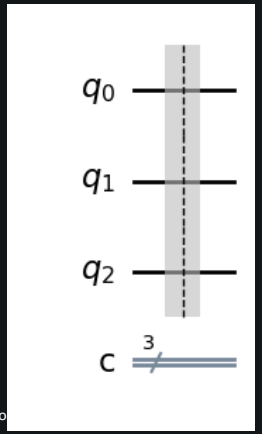
A.



B.



C.



D.

```
This solution raised the following error: 'Index 3 out of range for size 3.'
```

E.

```
This solution raised the following error: 'QuantumCircuit' object has no attribute 'barrier_all'
```

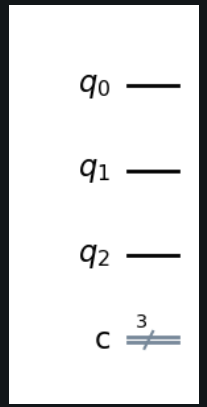
Question 11

11. Which two options would place a barrier across all qubits to the QuantumCircuit below?

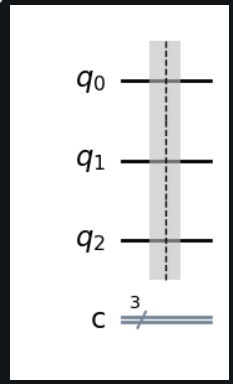
```
qc = QuantumCircuit(3,3)
```

A. `qc.barrier(qc)`
B. `qc.barrier([0,1,2])`
C. `qc.barrier()`
D. `qc.barrier(3)`
E. `qc.barrier_all()`

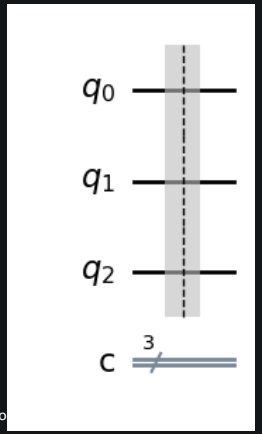
A.



B.



C.



D.

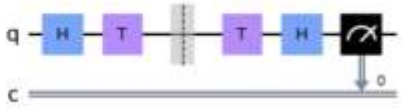
```
This solution raised the following error: 'Index 3 out of range for size 3.'
```

E.

```
This solution raised the following error: 'QuantumCircuit' object has no attribute 'barrier_all'
```

Question 12

12. What code fragment codes the equivalent circuit if you remove the barrier in the following QuantumCircuit?



A.
```
qc = QuantumCircuit(1,1)
qc.h(0)
qc.s(0)
qc.h(0)
qc.measure(0,0)
```

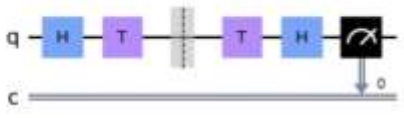
B.
```
qc = QuantumCircuit(1,1)
qc.measure(0,0)
```

C.
```
qc = QuantumCircuit(1,1)
qc.h(0)
qc.t(0)
qc.tdg(0)
qc.h(0)
qc.measure(0,0)
```

D.
```
qc = QuantumCircuit(1,1)
qc.h(0)
qc.z(0)
qc.h(0)
qc.measure(0,0)
```

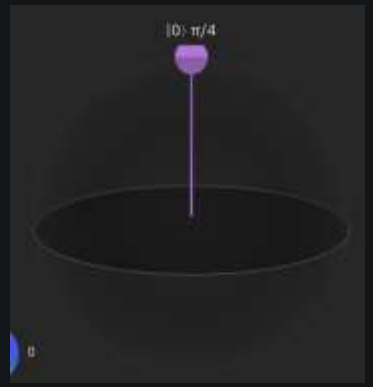
Question 12

12. What code fragment codes the equivalent circuit if you remove the barrier in the following QuantumCircuit?



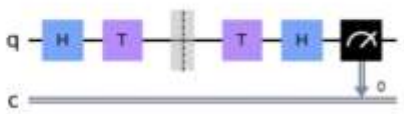
A.
```
qc = QuantumCircuit(1,1)
qc.h(0)
qc.s(0)
qc.h(0)
qc.measure(0,0)
```
B.
```
qc = QuantumCircuit(1,1)
qc.measure(0,0)
```
C.
```
qc = QuantumCircuit(1,1)
qc.h(0)
qc.t(0)
qc.tdg(0)
qc.h(0)
qc.measure(0,0)
```
D.
```
qc = QuantumCircuit(1,1)
qc.h(0)
qc.z(0)
qc.h(0)
qc.measure(0,0)
```

A.



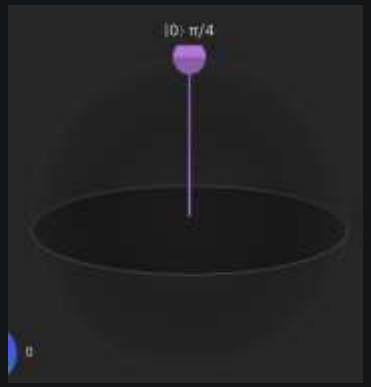
Question 12

12. What code fragment codes the equivalent circuit if you remove the barrier in the following QuantumCircuit?

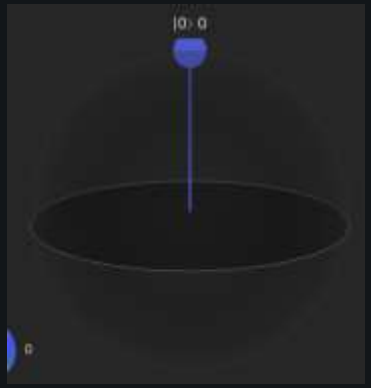


A.
```
qc = QuantumCircuit(1,1)
qc.h(0)
qc.s(0)
qc.h(0)
qc.measure(0,0)
```
B.
```
qc = QuantumCircuit(1,1)
qc.measure(0,0)
```
C.
```
qc = QuantumCircuit(1,1)
qc.h(0)
qc.t(0)
qc.tdg(0)
qc.h(0)
qc.measure(0,0)
```
D.
```
qc = QuantumCircuit(1,1)
qc.h(0)
qc.z(0)
qc.h(0)
qc.measure(0,0)
```

A.

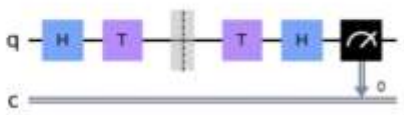


B.



Question 12

12. What code fragment codes the equivalent circuit if you remove the barrier in the following QuantumCircuit?



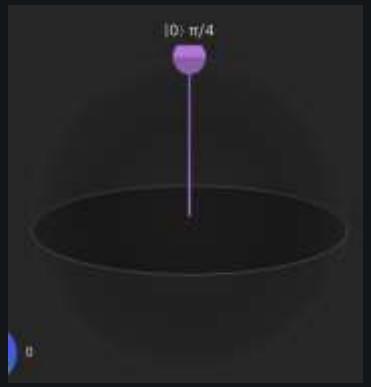
A.
```
qc = QuantumCircuit(1,1)
qc.h(0)
qc.s(0)
qc.h(0)
qc.measure(0,0)
```

B.
```
qc = QuantumCircuit(1,1)
qc.measure(0,0)
```

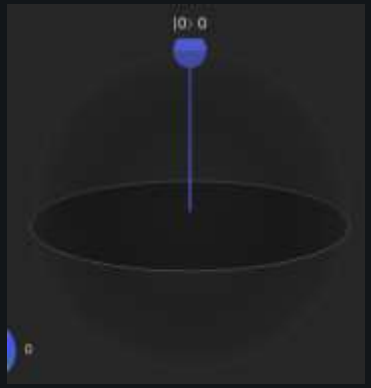
C.
```
qc = QuantumCircuit(1,1)
qc.h(0)
qc.t(0)
qc.tdg(0)
qc.h(0)
qc.measure(0,0)
```

D.
```
qc = QuantumCircuit(1,1)
qc.h(0)
qc.z(0)
qc.h(0)
qc.measure(0,0)
```

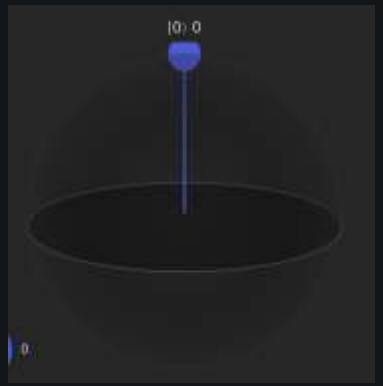
A.



B.

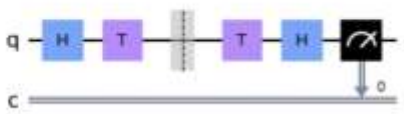


C.



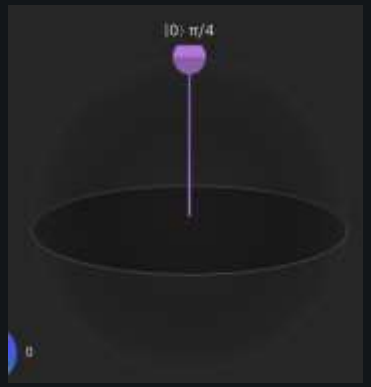
Question 12

12. What code fragment codes the equivalent circuit if you remove the barrier in the following QuantumCircuit?

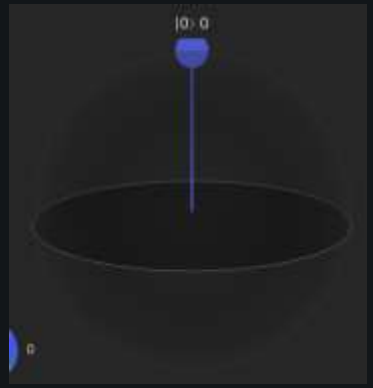


A.
```
qc = QuantumCircuit(1,1)
qc.h(0)
qc.s(0)
qc.h(0)
qc.measure(0,0)
```
B.
```
qc = QuantumCircuit(1,1)
qc.measure(0,0)
```
C.
```
qc = QuantumCircuit(1,1)
qc.h(0)
qc.t(0)
qc.tdg(0)
qc.h(0)
qc.measure(0,0)
```
D.
```
qc = QuantumCircuit(1,1)
qc.h(0)
qc.z(0)
qc.h(0)
qc.measure(0,0)
```

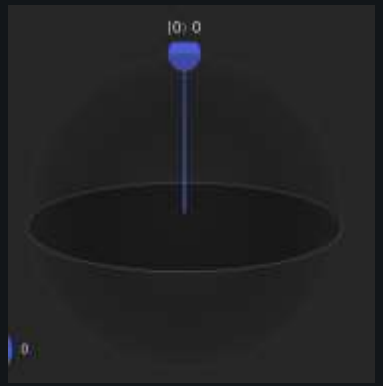
A.



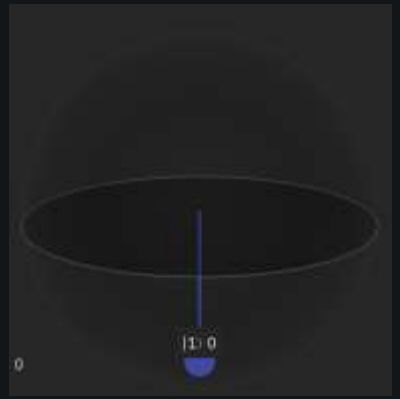
B.



C.



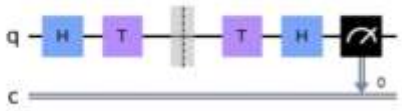
D.



Question 12

Qiskit

12. What code fragment codes the equivalent circuit if you remove the barrier in the following QuantumCircuit?



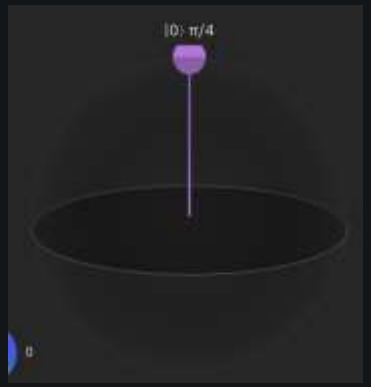
A.
```
qc = QuantumCircuit(1,1)
qc.h(0)
qc.s(0)
qc.h(0)
qc.measure(0,0)
```

B.
```
qc = QuantumCircuit(1,1)
qc.measure(0,0)
```

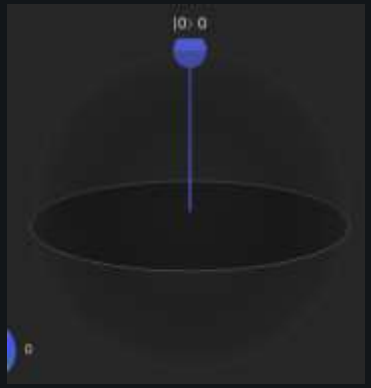
C.
```
qc = QuantumCircuit(1,1)
qc.h(0)
qc.t(0)
qc.tdg(0)
qc.h(0)
qc.measure(0,0)
```

D.
```
qc = QuantumCircuit(1,1)
qc.h(0)
qc.z(0)
qc.h(0)
qc.measure(0,0)
```

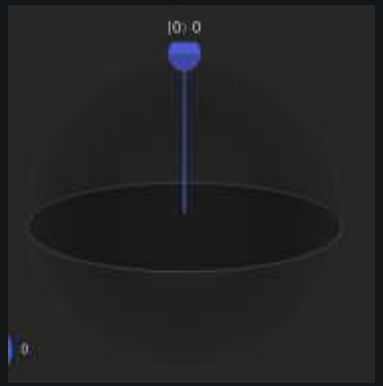
A.



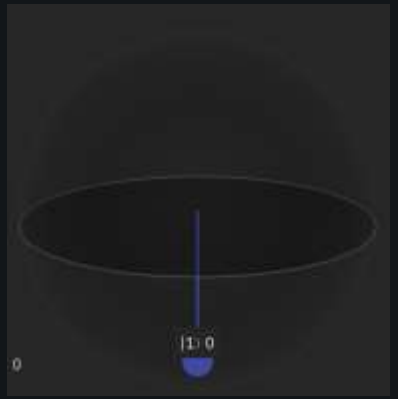
B.



C.



D.



Question 13

Qiskit

13. Given the following code, what is the depth of the circuit?

```
qc = QuantumCircuit(2, 2)

qc.h(0)
qc.barrier(0)
qc.cx(0,1)
qc.barrier([0,1])
```

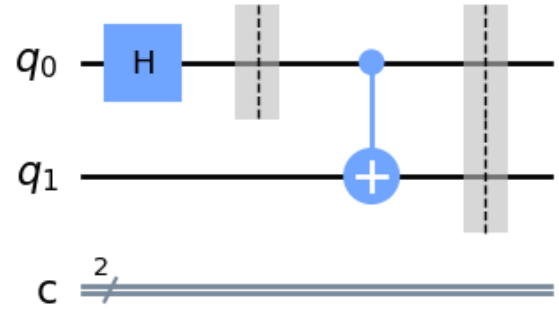
A. 2
B. 3
C. 4
D. 5

Question 13

13. Given the following code, what is the depth of the circuit?

```
qc = QuantumCircuit(2, 2)

qc.h(0)
qc.barrier(0)
qc.cx(0,1)
qc.barrier([0,1])
```

A. 2
B. 3
C. 4
D. 5



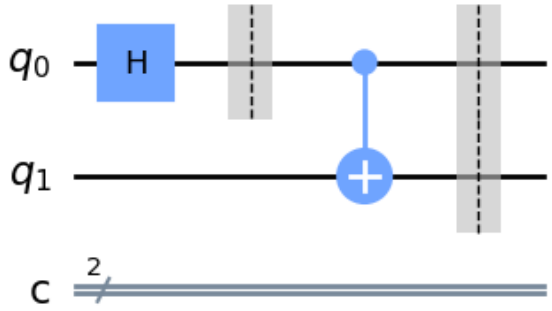
Question 13

Qiskit

13. Given the following code, what is the depth of the circuit?

```
qc = QuantumCircuit(2, 2)

qc.h(0)
qc.barrier(0)
qc.cx(0,1)
qc.barrier([0,1])
```

A. 2
B. 3
C. 4
D. 5



Question 14

Qiskit

14. Which code snippet would execute a circuit given these parameters?

1) • Measure the circuit 1024 times,
2) • use the QASM simulator,
3) • and use a coupling map that connects three qubits linearly

```
qc = QuantumCircuit(3)

# Insert code fragment here
result = job.result()
```

A.
```
qasm_sim = Aer.get_backend('qasm_simulator')
couple_map = [[0, 1], [1, 2]]
job = execute(qc, backend=qasm_sim, shots=1024,
coupling_map=couple_map)
```

B.
```
qasm_sim = Aer.getBackend('ibmq_simulator')
couple_map = [[0, 1], [0, 2]]
job = execute(qc, loop=1024, coupling_map=couple_map)
```

C.
```
qasm_sim = Aer.get_backend('qasm_simulator')
couple_map = [[0, 1], [1, 2]]
job = execute(qc, backend=qasm_sim, repeat=1024,
coupling_map=couple_map)
```

D.
```
qasm_sim = Aer.get_backend('qasm_simulator')
couple_map = [[0, 1], [1, 2]]
job = execute(backend=qasm_sim, qc, shot=1024,
coupling_map=couple_map)
```

Question 14

14. Which code snippet would execute a circuit given these parameters?

1) • Measure the circuit 1024 times,
2) • use the QASM simulator,
3) • and use a coupling map that connects three qubits linearly

```
qc = QuantumCircuit(3)

# Insert code fragment here
result = job.result()
```

A.
```
qasm_sim = Aer.get_backend('qasm_simulator')
couple_map = [[0, 1], [1, 2]]
job = execute(qc, backend=qasm_sim, shots=1024,
coupling_map=couple_map)
```
B.
```
qasm_sim = Aer.getBackend('ibmq_simulator')
couple_map = [[0, 1], [0, 2]]
job = execute(qc, loop=1024, coupling_map=couple_map)
```
C.
```
qasm_sim = Aer.get_backend('qasm_simulator')
couple_map = [[0, 1], [1, 2]]
job = execute(qc, backend=qasm_sim, repeat=1024,
coupling_map=couple_map)
```
D.
```
qasm_sim = Aer.get_backend('qasm_simulator')
couple_map = [[0, 1], [1, 2]]
job = execute(backend=qasm_sim, qc, shot=1024,
coupling_map=couple_map)
```

A.

```
Backend: qasm_simulator
Shots: 1024
```

Question 14

Qiskit

14. Which code snippet would execute a circuit given these parameters?

1) • Measure the circuit 1024 times,
2) • use the QASM simulator,
3) • and use a coupling map that connects three qubits linearly

```
qc = QuantumCircuit(3)

# Insert code fragment here
result = job.result()
```

A.
```
qasm_sim = Aer.get_backend('qasm_simulator')
couple_map = [[0, 1], [1, 2]]
job = execute(qc, backend=qasm_sim, shots=1024,
coupling_map=couple_map)
```

B.
```
qasm_sim = Aer.getBackend('ibmq_simulator')
couple_map = [[0, 1], [0, 2]]
job = execute(qc, loop=1024, coupling_map=couple_map)
```

C.
```
qasm_sim = Aer.get_backend('qasm_simulator')
couple_map = [[0, 1], [1, 2]]
job = execute(qc, backend=qasm_sim, repeat=1024,
coupling_map=couple_map)
```

D.
```
qasm_sim = Aer.get_backend('qasm_simulator')
couple_map = [[0, 1], [1, 2]]
job = execute(backend=qasm_sim, qc, shot=1024,
coupling_map=couple_map)
```

A.
```
Backend: qasm_simulator
Shots: 1024
```

B.
```
This code raised the following exception: 'AerProvider' object has no attribute 'getBackend'
```

Question 14

Qiskit

14. Which code snippet would execute a circuit given these parameters?

1) • Measure the circuit 1024 times,
2) • use the QASM simulator,
3) • and use a coupling map that connects three qubits linearly

```
qc = QuantumCircuit(3)

# Insert code fragment here
result = job.result()
```

A.
```
qasm_sim = Aer.get_backend('qasm_simulator')
couple_map = [[0, 1], [1, 2]]
job = execute(qc, backend=qasm_sim, shots=1024,
coupling_map=couple_map)
```

B.
```
qasm_sim = Aer.getBackend('ibmq_simulator')
couple_map = [[0, 1], [0, 2]]
job = execute(qc, loop=1024, coupling_map=couple_map)
```

C.
```
qasm_sim = Aer.get_backend('qasm_simulator')
couple_map = [[0, 1], [1, 2]]
job = execute(qc, backend=qasm_sim, repeat=1024,
coupling_map=couple_map)
```

D.
```
qasm_sim = Aer.get_backend('qasm_simulator')
couple_map = [[0, 1], [1, 2]]
job = execute(backend=qasm_sim, qc, shot=1024,
coupling_map=couple_map)
```

A.
```
Backend: qasm_simulator
Shots: 1024
```

B.
```
This code raised the following exception: 'AerProvider' object has no attribute 'getBackend'
```

C.
```
Backend: qasm_simulator
Repeats:  1024
```

Question 14

Qiskit

14. Which code snippet would execute a circuit given these parameters?

1) • Measure the circuit 1024 times,
2) • use the QASM simulator,
3) • and use a coupling map that connects three qubits linearly

```
qc = QuantumCircuit(3)

# Insert code fragment here
result = job.result()
```

A.
```
qasm_sim = Aer.get_backend('qasm_simulator')
couple_map = [[0, 1], [1, 2]]
job = execute(qc, backend=qasm_sim, shots=1024,
coupling_map=couple_map)
```

B.
```
qasm_sim = Aer.getBackend('ibmq_simulator')
couple_map = [[0, 1], [0, 2]]
job = execute(qc, loop=1024, coupling_map=couple_map)
```

C.
```
qasm_sim = Aer.get_backend('qasm_simulator')
couple_map = [[0, 1], [1, 2]]
job = execute(qc, backend=qasm_sim, repeat=1024,
coupling_map=couple_map)
```

D.
```
qasm_sim = Aer.get_backend('qasm_simulator')
couple_map = [[0, 1], [1, 2]]
job = execute(backend=qasm_sim, qc, shot=1024,
coupling_map=couple_map)
```

A.
```
Backend: qasm_simulator
Shots: 1024
```

B.
```
This code raised the following exception: 'AerProvider' object has no attribute 'getBackend'
```

C.
```
Backend: qasm_simulator
Repeats: 1024
```

D.
```
This raises a SyntaxError: invalid syntax (<string>, line 1)
```

Question 14

Qiskit

14. Which code snippet would execute a circuit given these parameters?

1) • Measure the circuit 1024 times,
2) • use the QASM simulator,
3) • and use a coupling map that connects three qubits linearly

```
qc = QuantumCircuit(3)

# Insert code fragment here
result = job.result()
```

A.
```
qasm_sim = Aer.get_backend('qasm_simulator')
couple_map = [[0, 1], [1, 2]]
job = execute(qc, backend=qasm_sim, shots=1024,
coupling_map=couple_map)
```

B.
```
qasm_sim = Aer.getBackend('ibmq_simulator')
couple_map = [[0, 1], [0, 2]]
job = execute(qc, loop=1024, coupling_map=couple_map)
```

C.
```
qasm_sim = Aer.get_backend('qasm_simulator')
couple_map = [[0, 1], [1, 2]]
job = execute(qc, backend=qasm_sim, repeat=1024,
coupling_map=couple_map)
```

D.
```
qasm_sim = Aer.get_backend('qasm_simulator')
couple_map = [[0, 1], [1, 2]]
job = execute(backend=qasm_sim, qc, shot=1024,
coupling_map=couple_map)
```

A.
```
Backend: qasm_simulator
Shots: 1024
```

B.
```
This code raised the following exception: 'AerProvider' object has no attribute 'getBackend'
```

C.
```
Backend: qasm_simulator
Repeats: 1024
```

D.
```
This raises a SyntaxError: invalid syntax (<string>, line 1)
```

Question 15

15. Which of these would execute a circuit on a set of qubits which are coupled in a custom way?

```
from qiskit import QuantumCircuit, execute, BasicAer
backend = BasicAer.get_backend('qasm_simulator')
qc = QuantumCircuit(3)

# insert code here
```

A. `execute(qc, backend, shots=1024, coupling_map=[[0,1], [1,2]])`
B. `execute(qc, backend, shots=1024, custom_topology=[[0,1],[2,3]]`
C. `execute(qc, backend, shots=1024, device="qasm_simulator", mode="custom")`
D. `execute(qc, backend, mode="custom")`

Question 15

15. Which of these would execute a circuit on a set of qubits which are coupled in a custom way?

```
from qiskit import QuantumCircuit, execute, BasicAer
backend = BasicAer.get_backend('qasm_simulator')
qc = QuantumCircuit(3)

# insert code here
```

A. `execute(qc, backend, shots=1024, coupling_map=[[0,1], [1,2]])`
B. `execute(qc, backend, shots=1024, custom_topology=[[0,1],[2,3]]`
C. `execute(qc, backend, shots=1024, device="qasm_simulator", mode="custom")`
D. `execute(qc, backend, mode="custom")`

Successfully run

Question 15

15. Which of these would execute a circuit on a set of qubits which are coupled in a custom way?

```
from qiskit import QuantumCircuit, execute, BasicAer
backend = BasicAer.get_backend('qasm_simulator')
qc = QuantumCircuit(3)

# insert code here
```

A. `execute(qc, backend, shots=1024, coupling_map=[[0,1], [1,2]])`
B. `execute(qc, backend, shots=1024, custom_topology=[[0,1],[2,3]]`
C. `execute(qc, backend, shots=1024, device="qasm_simulator", mode="custom")`
D. `execute(qc, backend, mode="custom")`

Successfully run

```
The following warning was raised: Option custom_topology is not used by this backen
```

Question 15

15. Which of these would execute a circuit on a set of qubits which are coupled in a custom way?

```
from qiskit import QuantumCircuit, execute, BasicAer
backend = BasicAer.get_backend('qasm_simulator')
qc = QuantumCircuit(3)

# insert code here
```

A. `execute(qc, backend, shots=1024, coupling_map=[[0,1], [1,2]])`
B. `execute(qc, backend, shots=1024, custom_topology=[[0,1],[2,3]]`
C. `execute(qc, backend, shots=1024, device="qasm_simulator", mode="custom")`
D. `execute(qc, backend, mode="custom")`

Successfully run

```
The following warning was raised: Option custom_topology is not used by this backen
```

```
The following warning was raised: Option device is not used by this backend
```

Question 15

15. Which of these would execute a circuit on a set of qubits which are coupled in a custom way?

```
from qiskit import QuantumCircuit, execute, BasicAer
backend = BasicAer.get_backend('qasm_simulator')
qc = QuantumCircuit(3)

# insert code here
```

A. `execute(qc, backend, shots=1024, coupling_map=[[0,1], [1,2]])`

B. `execute(qc, backend, shots=1024, custom_topology=[[0,1],[2,3]]`

C. `execute(qc, backend, shots=1024, device="qasm_simulator", mode="custom")`

D. `execute(qc, backend, mode="custom")`

Successfully run

```
The following warning was raised: Option custom_topology is not used by this backen
```

```
The following warning was raised: Option device is not used by this backend
```

```
The following warning was raised: Option mode is not used by this backend
```

Question 16

16. Which three simulators are available in BasicAer?

A. `qasm_simulator`
B. `basic_qasm_simulator`
C. `statevector_simulator`
D. `unitary_simulator`
E. `quantum_simulator`
F. `quantum_circuit_simulator`

Question 16

Qiskit

16. Which three simulators are available in BasicAer?

A. qasm_simulator
B. basic_qasm_simulator
C. statevector_simulator
D. unitary_simulator
E. quantum_simulator
F. quantum_circuit_simulator

In [2]:
```
for backend in BasicAer.backends():
    print(backend)
```

```
qasm_simulator
statevector_simulator
unitary_simulator
```

Question 16

Qiskit

16. Which three simulators are available in BasicAer?

A. qasm_simulator
B. basic_qasm_simulator
C. statevector_simulator
D. unitary_simulator
E. quantum_simulator
F. quantum_circuit_simulator

In [2]:
```
for backend in BasicAer.backends():
    print(backend)
```

```
qasm_simulator
statevector_simulator
unitary_simulator
```

Question 17

17. Which line of code would assign a statevector simulator object to the variable `backend` ?

A. `backend = BasicAer.StatevectorSimulatorPy()`
B. `backend = BasicAer.get_backend('statevector_simulator')`
C. `backend = BasicAer.StatevectorSimulatorPy().name()`
D. `backend = BasicAer.get_back('statevector_simulator')`

Question 17

17. Which line of code would assign a statevector simulator object to the variable `backend` **?**

A. `backend = BasicAer.StatevectorSimulatorPy()`
B. `backend = BasicAer.get_backend('statevector_simulator')`
C. `backend = BasicAer.StatevectorSimulatorPy().name()`
D. `backend = BasicAer.get_back('statevector_simulator')`

A.

```
Exception thrown: 'BasicAerProvider' object has no attribute 'StatevectorSimulatorPy'
```

Question 17

Qiskit

17. Which line of code would assign a statevector simulator object to the variable `backend` ?

A. `backend = BasicAer.StatevectorSimulatorPy()`
B. `backend = BasicAer.get_backend('statevector_simulator')`
C. `backend = BasicAer.StatevectorSimulatorPy().name()`
D. `backend = BasicAer.get_back('statevector_simulator')`

A.

```
Exception thrown: 'BasicAerProvider' object has no attribute 'StatevectorSimulatorPy'
```

C.

```
Exception thrown: 'BasicAerProvider' object has no attribute 'StatevectorSimulatorPy'
```

Question 17

Qiskit

17. Which line of code would assign a statevector simulator object to the variable `backend` ?

A. `backend = BasicAer.StatevectorSimulatorPy()`
B. `backend = BasicAer.get_backend('statevector_simulator')`
C. `backend = BasicAer.StatevectorSimulatorPy().name()`
D. `backend = BasicAer.get_back('statevector_simulator')`

A.
```
Exception thrown: 'BasicAerProvider' object has no attribute 'StatevectorSimulatorPy'
```

C.
```
Exception thrown: 'BasicAerProvider' object has no attribute 'StatevectorSimulatorPy'
```

D.
```
Exception thrown: 'BasicAerProvider' object has no attribute 'get_back'
```

Question 17

Qiskit

17. Which line of code would assign a statevector simulator object to the variable `backend` ?

A. `backend = BasicAer.StatevectorSimulatorPy()`
B. `backend = BasicAer.get_backend('statevector_simulator')`
C. `backend = BasicAer.StatevectorSimulatorPy().name()`
D. `backend = BasicAer.get_back('statevector_simulator')`

A.
```
Exception thrown: 'BasicAerProvider' object has no attribute 'StatevectorSimulatorPy'
```

C.
```
Exception thrown: 'BasicAerProvider' object has no attribute 'StatevectorSimulatorPy'
```

D.
```
Exception thrown: 'BasicAerProvider' object has no attribute 'get_back'
```

Question 18

18. Which code fragment would yield an operator that represents a single-qubit X gate?

A.
```
op = Operator.Xop(0)
```
B.
```
op = Operator([[0,1]])
```
C.
```
qc = QuantumCircuit(1)
qc.x(0)
op = Operator(qc)
```
D.
```
op = Operator([[1,0,0,1]])
```

Question 18

18. Which code fragment would yield an operator that represents a single-qubit X gate?

A.
```
op = Operator.Xop(0)
```
B.
```
op = Operator([[0,1]])
```
C.
```
qc = QuantumCircuit(1)
qc.x(0)
op = Operator(qc)
```
D.
```
op = Operator([[1,0,0,1]])
```

A.
```
Exception thrown: type object 'Operator' has no attribute 'Xop'
```

Question 18

18. Which code fragment would yield an operator that represents a single-qubit X gate?

A.
```
op = Operator.Xop(0)
```
B.
```
op = Operator([[0,1]])
```
C.
```
qc = QuantumCircuit(1)
qc.x(0)
op = Operator(qc)
```
D.
```
op = Operator([[1,0,0,1]])
```

A.
```
Exception thrown: type object 'Operator' has no attribute 'Xop'
```

B. Operator passed : $[0 \quad 1]$

Question 18

18. Which code fragment would yield an operator that represents a single-qubit X gate?

A.
```
op = Operator.Xop(0)
```
B.
```
op = Operator([[0,1]])
```
C.
```
qc = QuantumCircuit(1)
qc.x(0)
op = Operator(qc)
```
D.
```
op = Operator([[1,0,0,1]])
```

A.
```
Exception thrown: type object 'Operator' has no attribute 'Xop'
```

B. Operator passed : $[0 \quad 1]$

C. Operator passed : $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$

Question 18

Qiskit

18. Which code fragment would yield an operator that represents a single-qubit X gate?

A.
```
op = Operator.Xop(0)
```
B.
```
op = Operator([[0,1]])
```
C.
```
qc = QuantumCircuit(1)
qc.x(0)
op = Operator(qc)
```
D.
```
op = Operator([[1,0,0,1]])
```

A.
```
Exception thrown: type object 'Operator' has no attribute 'Xop'
```

B. Operator passed : $[0 \quad 1]$

C. Operator passed : $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$

D. Operator passed : $[1 \quad 0 \quad 0 \quad 1]$

Question 18

18. Which code fragment would yield an operator that represents a single-qubit X gate?

A.
```
op = Operator.Xop(0)
```
B.
```
op = Operator([[0,1]])
```
C.
```
qc = QuantumCircuit(1)
qc.x(0)
op = Operator(qc)
```
D.
```
op = Operator([[1,0,0,1]])
```

A.
```
Exception thrown: type object 'Operator' has no attribute 'Xop'
```

B. Operator passed : $[0 \quad 1]$

C. Operator passed : $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$

D. Operator passed : $[1 \quad 0 \quad 0 \quad 1]$

Question 19

19. What would be the fidelity result(s) for these two operators, which differ only by global phase?

```
op_a = Operator(XGate())
op_b = numpy.exp(1j * 0.5) * Operator(XGate())
```

A. `state_fidelity()` of 1.0
B. `state_fidelity()` and `average_gate_fidelity()` of 1.0
C. `average_gate_fidelity()` and `process_fidelity()` of 1.0
D. `state_fidelity()`, `average_gate_fidelity()` and `process_fidelity()` of 1.0

Question 19

19. What would be the fidelity result(s) for these two operators, which differ only by global phase?

```
op_a = Operator(XGate())
op_b = numpy.exp(1j * 0.5) * Operator(XGate())
```

A. `state_fidelity()` of 1.0
B. `state_fidelity()` and `average_gate_fidelity()` of 1.0
C. `average_gate_fidelity()` and `process_fidelity()` of 1.0
D. `state_fidelity()`, `average_gate_fidelity()` and `process_fidelity()` of 1.0

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

$$op_b : \begin{bmatrix} 0 & 0.8775825619 + 0.4794255386i \\ 0.8775825619 + 0.4794255386i & 0 \end{bmatrix}$$

Average gate fidelity measures how close two quantum operations (or gates) are, on average, for all possible input states.
Process fidelity is a measure of the similarity between two quantum processes or channels. It's a more general concept than state fidelity, which looks at the entire process of transformation of a quantum state.

Question 19

19. What would be the fidelity result(s) for these two operators, which differ only by global phase?

```
op_a = Operator(XGate())
op_b = numpy.exp(1j * 0.5) * Operator(XGate())
```

A. state_fidelity() of 1.0
B. state_fidelity() and average_gate_fidelity() of 1.0
C. average_gate_fidelity() and process_fidelity() of 1.0
D. state_fidelity(), average_gate_fidelity() and process_fidelity() of 1.0

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

$$op_b : \begin{bmatrix} 0 & 0.8775825619 + 0.4794255386i \\ 0.8775825619 + 0.4794255386i & 0 \end{bmatrix}$$

Average gate fidelity measures how close two quantum operations (or gates) are, on average, for all possible input states.
Process fidelity is a measure of the similarity between two quantum processes or channels. It's a more general concept than state fidelity, which looks at the entire process of transformation of a quantum state.

Question 20

20. Given this code fragment, which output fits most closely with the measurement probability distribution?

```
qc = QuantumCircuit(2, 2)
qc.x(0)
qc.measure([0,1], [0,1])
simulator = Aer.get_backend('qasm_simulator')
result = execute(qc, simulator, shots=1000).result()
counts  = result.get_counts(qc)
print(counts)
```

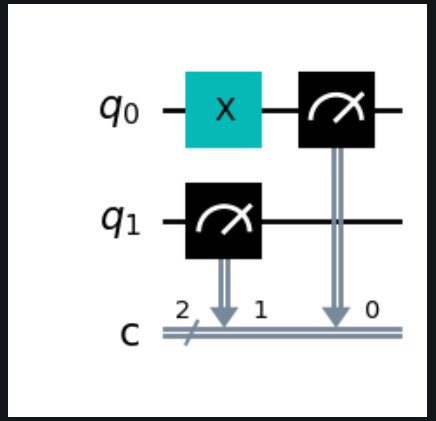
A. {'00': 1000}
B. {'01': 1000}
C. {'10': 1000}
D. {'11': 1000}

Question 20

20. Given this code fragment, which output fits most closely with the measurement probability distribution?

```
qc = QuantumCircuit(2, 2)
qc.x(0)
qc.measure([0,1], [0,1])
simulator = Aer.get_backend('qasm_simulator')
result = execute(qc, simulator, shots=1000).result()
counts  = result.get_counts(qc)
print(counts)
```

A. {'00': 1000}
B. {'01': 1000}
C. {'10': 1000}
D. {'11': 1000}

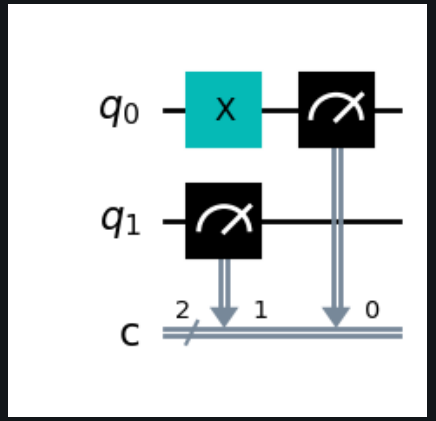


Question 20

20. Given this code fragment, which output fits most closely with the measurement probability distribution?

```
qc = QuantumCircuit(2, 2)
qc.x(0)
qc.measure([0,1], [0,1])
simulator = Aer.get_backend('qasm_simulator')
result = execute(qc, simulator, shots=1000).result()
counts  = result.get_counts(qc)
print(counts)
```

A. {'00': 1000}
B. {'01': 1000}
C. {'10': 1000}
D. {'11': 1000}



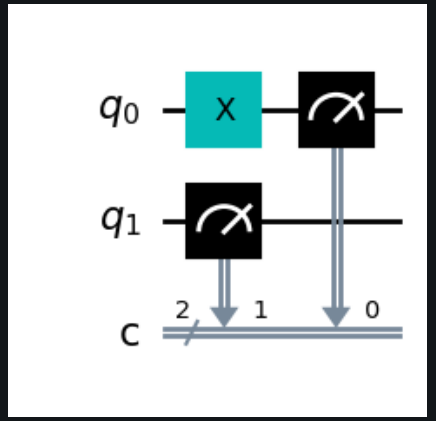

```
In [3]: simulator = Aer.get_backend("qasm_simulator")
        result = execute(qc, simulator, shots=1000).result()
        counts = result.get_counts(qc)
        print(counts)

{'01': 1000}
```

Question 20

20. Given this code fragment, which output fits most closely with the measurement probability distribution?

```
qc = QuantumCircuit(2, 2)
qc.x(0)
qc.measure([0,1], [0,1])
simulator = Aer.get_backend('qasm_simulator')
result = execute(qc, simulator, shots=1000).result()
counts  = result.get_counts(qc)
print(counts)
```

A. {'00': 1000}
B. {'01': 1000}
C. {'10': 1000}
D. {'11': 1000}




```
In [3]: simulator = Aer.get_backend("qasm_simulator")
        result = execute(qc, simulator, shots=1000).result()
        counts = result.get_counts(qc)
        print(counts)

{'01': 1000}
```