IBM Professional Certification Program	**Exam C1000 – 112 IBM Certified Associate Developer - Qiskit v0.2X**

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])`

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

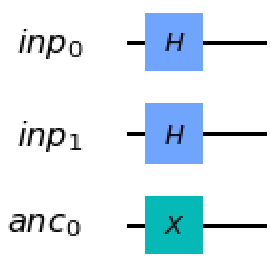
```
qc = QuantumCircuit(1)
qc.ry(3 * math.pi/4, 0)
```

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

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

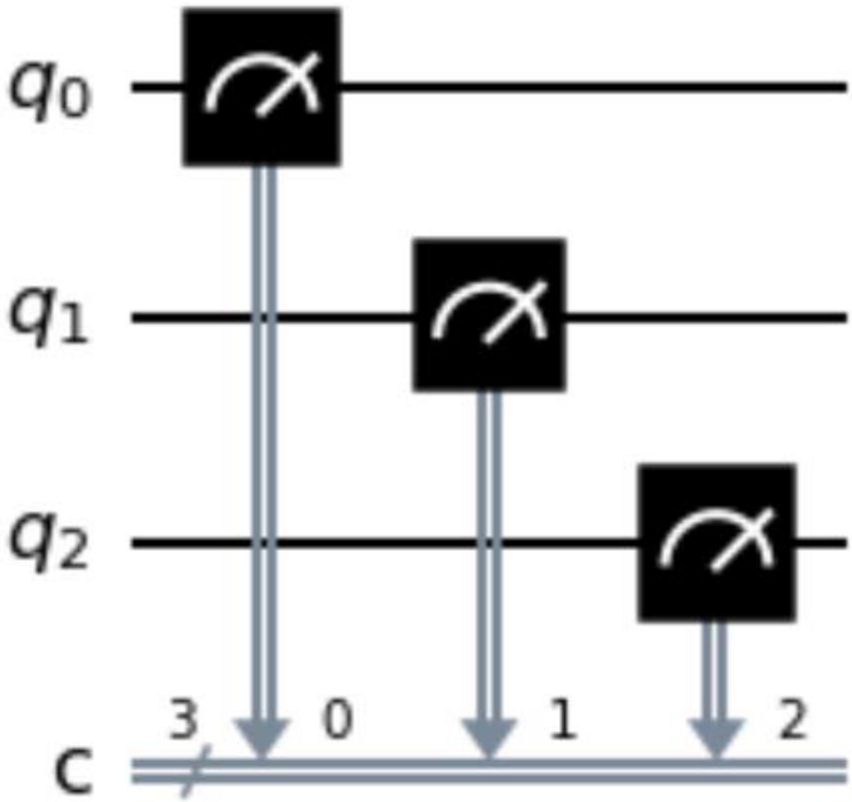
```
inp_reg = QuantumRegister(2, name='inp')
ancilla = QuantumRegister(1, name='anc')
qc = QuantumCircuit(inp_reg, ancilla)

# 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()
```

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)`

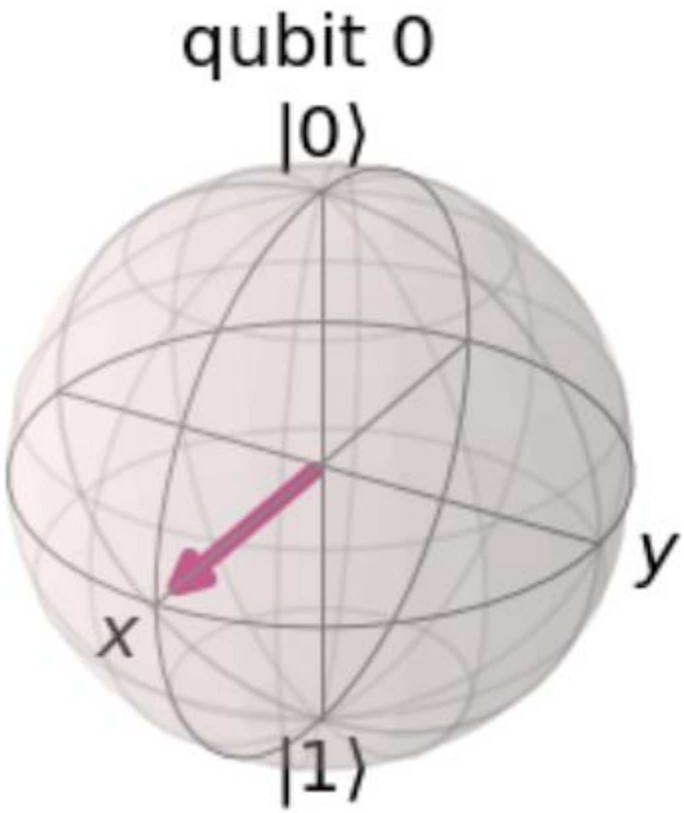
5. **Which code fragment will produce a maximally entangled, or Bell, state?**

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)
```

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)`

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

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()
```

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)`

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

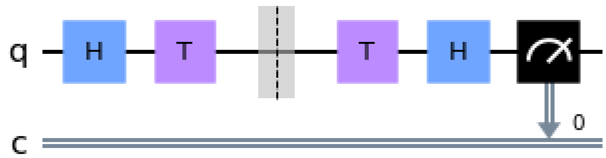
   A. `qc.ccx(0,1,2)`
   B. `qc.mct([0,1], 2)`
   C. `from qiskit.circuit.library import CXGate`
```
ccx = CXGate().control()
qc.append(ccx, [0,1,2])
```
   D. `qc.cry(0,1,2)`

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()`

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)
```

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

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)
```

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")`

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`

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')`

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]])`

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`

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}`

Answer Key:

1) A
2) C
3) A,B,D
4) A
5) A
6) A,C
7) B
8) A,B
9) D
10) D
11) B,C
12) A
13) A
14) A
15) A
16) A,C,D
17) B
18) C
19) C
20) B