Name: Okoye Adunife Kizito
Student ID: 100611918
CSCI 4140U

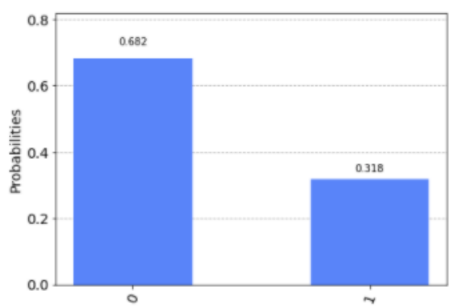
Laboratory Six

Ten gates is a relatively small circuit. Try the same code with N=10. **Cut and paste the resulting histogram into your report.**

In [30]:
```
def get_noise(p_meas, p_gate):
    error_meas = pauli_error([('X', p_meas), ('I', 1 - p_meas)])
    error_gate1 = depolarizing_error(p_gate, 1)
    error_gate2 = error_gate1.tensor(error_gate1)

    noise_model = NoiseModel()
    noise_model.add_all_qubit_quantum_error(error_meas, "measure") # measurement error is applied to measurements
    noise_model.add_all_qubit_quantum_error(error_gate1, ["x"]) # single qubit gate error is applied to x gates
    noise_model.add_all_qubit_quantum_error(error_gate2, ["cx"]) # two qubit gate error is applied to cx gates

    return noise_model

noise_model = get_noise(0.01,0.05)
N=10

qc3=QuantumCircuit(1,1)
for i in range(N):
    qc3.x(0)
    qc3.x(0)
qc3.measure(qc3.qregs[0], qc3.cregs[0])
counts = execute(qc3, backend, noise_model=noise_model).result().get_counts()

print(counts)
backend = Aer.get_backend('qasm_simulator')
shots = 2048

plot_histogram(counts)
```

```
{'0': 698, '1': 326}
```

Out[30]:

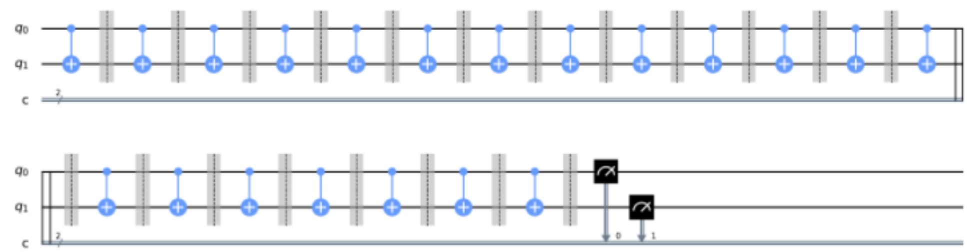


In this case we are getting a significant amount of error. Even the first bit, which has the value |1> throughout the circuit has its value changed over 20% of the time. Increase the value of N to 10 and run the circuit. **Cut and paste the results into your report. This doesn't look very promising.**

In [34]:
```
def get_noise(p_meas, p_gate):
    error_meas = pauli_error([('X', p_meas), ('I', 1 - p_meas)])
    error_gate1 = depolarizing_error(p_gate, 1)
    error_gate2 = error_gate1.tensor(error_gate1)

    noise_model = NoiseModel()
    noise_model.add_all_qubit_quantum_error(error_meas, "measure") # measurement error is applied to measurements
    noise_model.add_all_qubit_quantum_error(error_gate1, ["x"]) # single qubit gate error is applied to x gates
    noise_model.add_all_qubit_quantum_error(error_gate2, ["cx"]) # two qubit gate error is applied to cx gates

    return noise_model

noise_model = get_noise(0.01,0.05)
N=10

qc4=QuantumCircuit(2,2)
for i in range(N):
    qc4.cx(0,1)
    qc4.barrier()
    qc4.cx(0,1)
    qc4.barrier()

qc4.measure(qc4.qregs[0], qc4.cregs[0])
counts = execute(qc4, backend, noise_model=noise_model).result().get_counts()

print(counts)
plot_histogram(counts)
qc4.draw('mpl')
```

```
{'00': 410, '01': 174, '10': 251, '11': 189}
```

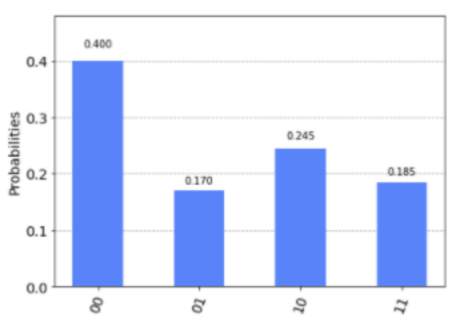
Out[34]:



In [35]:
```
print(counts)
backend = Aer.get_backend('qasm_simulator')
shots = 2048

plot_histogram(counts)
```

```
{'00': 410, '01': 174, '10': 251, '11': 189}
```

Out[35]:



Note that we have a very high probability of measuring the correct result. We can do the same thing with the qubits initialized to |1>. This requires an extra gate for each qubit, so the probability of the correct result will be lower. For the report take the |1> case and increase the number of qubits to 5. How does this impact the probability of getting the correct answer? **Cut and paste your results into your laboratory report.**

In [25]:
```
noise_model = get_noise(0.01, 0.01)

qc0 = QuantumCircuit(5,5,name='0') # initialize circuit with three qubits in the 0 state

qc0.measure(qc0.qregs[0],qc0.cregs[0]) # measure the qubits

# run the circuit with th noise model and extract the counts
backend = Aer.get_backend('qasm_simulator')
counts = execute( qc0, backend,noise_model=noise_model).result().get_counts()

print(counts)
plot_histogram(counts)
```

```
{'00000': 970, '00001': 19, '10000': 8, '10010': 1, '00010': 5, '00100': 11, '01000': 9, '01001': 1}
```

Out[25]: