**CS3650 Final Project – Quantum Computer**

**Name: Nhan Thai**

**Github link: https://github.com/NhanThai0501/CS3650-Final-Project.git**

**Printout of code**

|  |
| --- |
| from qiskit import QuantumRegister, ClassicalRegister  from qiskit import QuantumCircuit, Aer, execute  from qiskit.tools.visualization import plot\_histogram  from IPython.core.display import display  print("Project: QUANTUM COIN TOSS\n")  print("\nPart 1: One quantum coins toss")  # toss 1 time  print("Number of tossing time: 1")  qBit = QuantumRegister(1)  cBit = ClassicalRegister(1)  qc = QuantumCircuit(qBit, cBit)  qc.h(qBit[0])  qc.measure(qBit, cBit)  display(qc.draw('mpl'))  print(qc)  display(qc.draw('text'))  backend = Aer.get\_backend('qasm\_simulator')  job = execute(qc, backend, shots = 1)  result = job.result()  counts = result.get\_counts(qc)  print(counts)  display(plot\_histogram(counts)) |
| # toss 10 times  print("Number of tossing time: 10")  qc.h(qBit[0])  qc.measure(qBit, cBit)  display(qc.draw('mpl'))  print(qc)  display(qc.draw('text'))  backend = Aer.get\_backend('qasm\_simulator')  job = execute(qc, backend, shots = 10)  result = job.result()  counts = result.get\_counts(qc)  print(counts)  display(plot\_histogram(counts))  # toss 1000 times  print("Number of tossing time: 1000")  qc.h(qBit[0])  qc.measure(qBit, cBit)  display(qc.draw('mpl'))  print(qc)  display(qc.draw('text'))  backend = Aer.get\_backend('qasm\_simulator')  job = execute(qc, backend, shots = 1000)  result = job.result()  counts = result.get\_counts(qc)  print(counts)  display(plot\_histogram(counts))  # toss 10000 times  print("Number of tossing time: 10000")  qc.h(qBit[0])  qc.measure(qBit, cBit)  display(qc.draw('mpl'))  print(qc)  display(qc.draw('text'))  backend = Aer.get\_backend('qasm\_simulator')  job = execute(qc, backend, shots = 10000)  result = job.result()  counts = result.get\_counts(qc)  print(counts)  display(plot\_histogram(counts))  # toss 20000 times  print("Number of tossing time: 20000")  qc.h(qBit[0])  qc.measure(qBit, cBit)  display(qc.draw('mpl'))  print(qc)  display(qc.draw('text'))  backend = Aer.get\_backend('qasm\_simulator')  job = execute(qc, backend, shots = 20000)  result = job.result()  counts = result.get\_counts(qc)  print(counts)  display(plot\_histogram(counts)) |
| print("Part 2: Two quantum coins toss")  qc = QuantumCircuit(2, 2)  qc.h([0,1])  qc.measure([0,1],[0,1])  display(qc.draw('mpl'))  backend = Aer.get\_backend('qasm\_simulator')  counts = execute(qc, backend, shots = 1000).result().get\_counts(qc)  display(plot\_histogram(counts)) |
| print("Part 3: Two quantum coins toss with CX gate")  qc = QuantumCircuit(2, 2)  qc.h(0)  qc.cx(0,1)  qc.measure([0,1],[0,1])  display(qc.draw('mpl'))  backend = Aer.get\_backend('qasm\_simulator')  counts = execute(qc, backend, shots = 1000).result().get\_counts(qc)  display(plot\_histogram(counts)) |
| print("Part 4: Three quantum coins toss")  qc = QuantumCircuit(3, 3)  qc.h([0,1,2])  qc.measure([0,1,2],[0,1,2])  display(qc.draw('mpl'))  backend = Aer.get\_backend('qasm\_simulator')  counts = execute(qc, backend, shots = 1000).result().get\_counts(qc)  display(plot\_histogram(counts))  qc.barrier([0,1,2])  qc.reset([0,1,2])  qc.h(0)  qc.cx(0,1)  qc.cx(0,2)  qc.measure([0,1,2],[0,1,2])  display(qc.draw('mpl'))  counts = execute(qc, backend, shots = 1000).result().get\_counts(qc)  display(plot\_histogram(counts)) |

**Output result**

|  |
| --- |
| **Cell #1** |
|  |
| **Cell #2** |
|  |
| **Cell #3** |
|  |
| **Cell #4** |
|  |
| **Cell #5** |
|  |