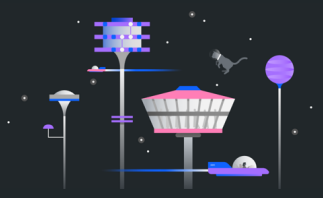
Qiskit
Fall Fest 2024



Hackathon Prompt:
Quantum Random Number Generator

Background and motivation:

Did you know that random number generators are a valuable tool for computing? Real random number generation is incredibly important for things like security and cryptography. However, all "random number generators" from classical computers aren't really.... random.

Your prompt is to create a truly random number generator using a quantum processor.

Getting started:

Your task is to create a random number generating circuit. You can decide on the number of qubits and possible outcomes, we'll call this number *n*. Keep in mind that creating a superposition chain of more than 10 qubits can be quite tricky.

Utilizing H-gates, generate a quantum state with *n* number of equally possible outcomes. How many qubits are needed for this task? It may be helpful to think about how the quantum computational space scales with each additional qubit. Assign your numeric values to the possible states, so that the measured state of the qubits results in your program printing the associated value.

Tips:

You will be able to decide how many times your quantum circuit is run by defining the number of "shots." By running your circuit once, you can create the quantum state with *n* equally possible outcomes. By only measuring once, the first outcome to be measured will be your answer. However, this doesn't fully demonstrate the fact that your qubits are in a quantum state of superposition. If you run it multiples times, you should get *n* equal buckets of measurement results, showing that each quantum state was equally likely to be measured. Now you can see the true quantum nature of your circuit, but you aren't given a clear result for your random number generator. You can decide how many "shots" you want to run, and, if you run it more than once, how you want your program to then randomly choose the quantum state to generate a number.

Deeper questions:

As you run more and more "shots," you might find that some numbers will be returned more often than others. This is a result of intrinsic noise in the quantum system. Your next challenge is to figure out how to mitigate this noise.

You can explore certain questions like, why some values are more likely to be returned than others? What is the best error mitigation technique (there are quite a few!) to implement to make this random number generator as "fair" as possible?

Suggested resources:

- Basics of Quantum Information
- Quantum Magic Eight Ball
- Qiskit Fall Fest 2024 Notebook 1
- Error Mitigation and Suppression Techniques Documentation

Proposed Project Judging Criteria

1. **Technical Aspects** (30 total points)

How complex is the quantum algorithm? Is it well optimized? Can the architecture serve users at a reasonable scale? How accessible is the end user application? Is it easy to use and intuitive for end users? Did the team use any significant parts of the Qiskit SDK, Qiskit Runtime, or other parts of the Qiskit ecosystem?

2. **Originality and Uniqueness** (25 total points)

How unique is this project compared to others? How interesting is it? Did the team attempt something new or difficult?

3. **Usefulness and Complexity** (25 total points)

How useful is the project and how well-designed is it? How functional is it at the time of judging? Can it be used in real-world business applications or serve as a valuable tool for individuals? Are there ways this project could be further built out and refined upon?

4. **Presentation** (20 total points)

How well did the team present their project? Were they able to explain their decisions? Did the entire team have a chance to speak? Did they tell a cohesive story?