Project Report

Project Design

Complex.Java

- The complex java file represents complex numbers in the form of x+y*i in which x and y are real numbers
- This class was an open course class available online
- In the case of having imaginary numbers in the exponent of a number (such as e^I*theta) I used euler's formula to expand out the expression in which the imaginary component was representable as a factor

Gates.Java

- This class stored all the gates as static matrices of Complex numbers
- Also stored pi and e as constants

Observable.java

- The Observable class represents a matrix or vector
- The class contains standard methods that allow access and some level of modification
- This constructor takes in a matrix or a string name of a gate and fills the instance variable of a complex number matrix accordingly using *Gates.Java*
- The class also contains a specific method that tensors two vectors together which is necessary to compute the probabilities at the end of the circuit

Simulator.java

- The constructor of this class takes in a string of the input qasm file and sets up the method computeStage()
- computeStage() is the main method of this class which is a recursive method that computes each gate in the circuit
- calcProbs() simply tensors the existing vectors together, and adds up the separate tensor products to compute the probabilities. The indices of the tensored vector correspond with a bitstring (index 6 is 110)
- There are a couple methods for the sake of making a deep copy of an observable and evaluating string expressions for unitaries
- handleCnot() separates the existing array of observables into 2, based on the probabilities of |0> and |1> in the control qubit
- getUnitary() parses through the line that represents a unitary gate. Converts the string into a matrix with a parser.

TextRunner.java

- This runs the simulator, and calculates the number of shots based on the probabilities computed by the Simulator

Using the Simulator

- The Simulator requires an input of the number of shots which must be a power of 2

- A file named “TestStuff.qasm” must be in the same directory as the jar executable in order for the application to run

Major Problems and Solutions

- Problem: The biggest issue was implementing the cnot. The data structure stores the vector of each qubit separately which contradicts the idea of entanglement
- Solution: Separating the array into two components, one containing the probability of |0> in the control qubit, and one containing the probability of |1> in the control qubit along with the X gate applied on the proper qubit.
- Problem: Calculating probabilities was more difficult in the case of cnot, as there were multiple sets of vectors being summed together.
- Solution: Tensor each set of vectors and add the tensored vectors together