Quantum Computation & Cryptography
Lab sheet 4

Quantum error correction

1. Implement the bit-flip error-correcting code described in the lectures. I.e. implement the following functions:

   (a) *xflipEncoder (qs:Qubits) (idxes:List<int>)* which takes a list of qubits and a list of 3 indices and does the following: encodes the qubit given by the first index (*idxes.[0]*), call it $|\psi\rangle = a|0\rangle + b|1\rangle$ as $|\psi\rangle_L = a|000\rangle + b|111\rangle$ (where the other 2 qubits are given by the other two indices).

   (b) *xflipDecoder (qs:Qubits) (idxes:List<int>)* does the opposite of the previous function, decoding $|\psi\rangle_L$ to $|\psi\rangle|0\rangle|0\rangle$.

   (c) *xflipCorrecter (qs:Qubits)* which takes a list of qubits (assumed to represent one logical qubit $|\psi\rangle_L$), performs a syndrome measurement and applies the appropriate correction. You are given the syndrome measurement function (*xflipSyndrome (qs:Qubits)*).

   (d) *testXFlip (qs:Qubits) (n:int)* which takes a list of qubits (assumed to represent one logical qubit $|\psi\rangle_L$) and a number of iterations and successively applies an X error on a randomly chosen qubit and then performs the correction. Print the state before and after to see if the correction procedure works.

2. Implement a phase-flip error-correcting code. This works just like the bit-flip code, but corrects for Z errors instead of X errors. The way it works is that the state $|\psi\rangle = a|0\rangle + b|1\rangle$ is encoded as $|\psi\rangle_L = a|+++\rangle + b|---\rangle$ instead of $|\psi\rangle_L = a|000\rangle + b|111\rangle$, and the same ideas from before apply here as well (with Z flips for correction instead of X). Implement the same functions as before with the prefix $pflip$ and additionally the *xflipSyndrome (qs:Qubits)* syndrome measurement function.

3. Time to compose the previous codes. We take a qubit $|\psi\rangle$ and encode it in the bit flip code. Now we take each (of the 3) qubit of this new state and encode them in the phase flip code. The resulting 9 qubit state should look like this:

   $$|\psi\rangle_L = \frac{1}{2\sqrt{2}}(a(|000\rangle+|111\rangle)(|000\rangle+|111\rangle)(|000\rangle+|111\rangle)+b(|000\rangle-|111\rangle)(|000\rangle-|111\rangle)(|000\rangle-|111\rangle))$$

   We can detect bit flip errors by performing the *xflip* syndrome measurement on each group of 3 qubits. Analougsly, we can detect phase flip errors among the groups of 3 qubits. This is called **Shor's code**.

It has the ability to detect and correct for *any* single qubit errors (including measurements). You are already given the functions to do this (they require the implementation of the previous two codes). Run the testing function *testShor*.

4. The syndrome measurements of the previous code seemed quite artificial. We would like to implement the *xflip* measurement using only standard gates and computational basis measurement. Write a function *parityCheck (qs:Qubits) (anc:Qubits)* which first takes a list of 3 qubits representing an encoded qubit, $|\psi\rangle_L$ in the bit flip code and a second list of 2 ancilla qubits. Make it so that the first ancilla qubit will store the parity of the first two encoded qubits and the second ancilla stores the parity of the second and third encoded qubits. Measure the ancillas for different encoded states and compare the results with those of *xflipSyndrome*.

5. All the operations done so far primarily used CNOT, H, X, Z and computational basis measurements. The function *bigOperation (qs:Qubits)* is simply a large quantum circuit using these gates. You are given 3 functions which test the behaviour of *bigOperation*. These functions are *naiveBigOperation*, *optimizedBigOperation* and *stabilizerBigOperation*. They simply run the *bigOperation* function many times and gather statistics about the measurement result of the last qubit. Test each of these functions with a certain number of qubits and say 100 iterations. What do you notice? The reason *stabilizerBigOperation* is able to perform the computation so fast is because of a result called the Gottesman-Knill theorem[1]. The theorem states that quantum circuits comprising of so-called Clifford gates (consisting of CNOT, H, X, Y, Z and others) and computational basis measurements can be simulated efficiently classically (in time $O(NlogN)$ where N is the number of quantum gates).

[1] http://en.wikipedia.org/wiki/Gottesman%E2%80%93Knill_theorem