CSCI-B490: Quantum Programming
Homework 3
Due: Tues, Feb 4

This assignment will be submitted to Canvas under Homework 3. You'll upload a pdf document named `hw3.pdf` containing your written responses. You'll also upload your qasm files. For the sanity of the graders, name your code files using the following format: `hw3-exE-P.qasm` or `hw3-exE-P.py` where `E` is the exercise number and `P` is the part number.

It is not currently possible to define a circuit in Open QASM which is parametrized by the number of input bits. For example, we know that for all $n > 0$, there exists an n-bit circuit which negates every input. We might call such a circuit `x`(n). On paper, we can define such things recursively in pidgin Python:

```
def x(n):
  if n = 1:
    return single-bit circuit with an x gate
  if n > 1:
    c = x(n-1)
    add a new register q to c
    apply x to q
    return c
```

Exercise 1. (*10 points*)

1. Design a 5-bit ripple adder (cf. Lecture on 1/28)
2. Show how to construct for all $n > 0$, `adder`(n), in pidgin python.
3. Define an actual python function `adder_circuit` which takes a positive integer n and returns a QuantumCircuit implementing `adder`(n). (QuantumCircuit is a class you can import from qiskit. QuantumRegister is also a useful class here.)

   Next write a wrapper `plus` which takes two non-negative integers and computes their sum by running `adder_circuit` on appropriate inputs.

Exercise 2. (*12 points*)

1. Convert truth tables for the following functions/circuits into cyclic notation:
   - `x` (in a 1-bit context)
   - `x` (in a 4-bit context)
   - `cx` (in a 2-bit context)
   - `cx` (in a 4-bit context)
   - `ccx` (in a 3-bit context)
   - `ccx` (in a 4-bit context)
2. Recall this theorem discussed in lecture:

   Theorem. *Let σ be a permutation and let*

   $$\sigma = (a_1\ b_1)(a_2\ b_2)\cdots(a_m\ b_m)$$
   $$\sigma = (c_1\ d_1)(c_2\ d_2)\cdots(c_n\ d_n)$$

   be two decompositions of σ into transpositions. Then m is even iff n is even.

   Using this theorem, prove that the function CCCNOT cannot be implemented with a 4-bit circuit using the gate basis $\{\texttt{x}, \texttt{cx}, \texttt{ccx}\}$.
3. Prove also that a 5-bit majority function cannot be implemented with a 5-bit circuit using the gate basis $\{\texttt{x}, \texttt{cx}, \texttt{ccx}\}$.

Exercise 3. (*2 points*). Define `toffoli`(n) in Π (cf. Lecture on 1/30).

Exercise 4. (*2 points*). Define `toffoli`(n) in Theseus (cf. Lecture on 1/30).

Exercise 5. (*4 extra credit points*). Use Quipper (cf. Lecture on 1/30) to generate a reversible n-bit multiplier circuit.