CSE 190 / Math 152 - Introduction to Quantum Computing
Homework 5
Due **Tuesday, May 21st, 1:30pm**

Instructions: **You may work individually or in a team of 2 people.** You may switch teams for different assignments. Please ensure your name(s) and PID(s) are clearly visible on the first page of your submission, and then upload the PDF to Gradescope. If working in a group, submit only one submission per group: one partner uploads the submission through their Gradescope account and then adds the other group member to the Gradescope submission by selecting their name in the "Add Group Members" dialog box. You will need to re-add your group member every time you resubmit a new version of your assignment.

It is highly recommended (though not required) that you type your answers. It is your responsibility to make any handwriting clear and legible for grading. A LaTeX template for the homework is provided on Canvas.

We will only be grading some of the problems below for correctness. However, because all of the concepts are important, we will not reveal which problems are being graded for correctness until after the assignment has been submitted. The remaining problems will be graded for completeness (i.e., does it look like there was a good-faith effort to solve the problem?).

Problems:

In this class, our goal has always been to show there's some problem that a quantum computer can solve efficiently that looks out of reach for classical computers. But, is it possible that the reverse is also true? That is, is there a problem that a classical computer could solve efficiently that cannot be solved efficiently with a quantum computer? In this problem, we will show that the answer is "no".

First, let's explore why the answer might be "yes". Namely, a classical circuit has access to gates that do not need to be unitary. A classical gate $g\colon \{0,1\}^c \to \{0,1\}$ is just a Boolean function which maps c bits to 1 bit. For example, the classical AND gate is the 2-input gate such that the output bit is 1 if and only if both of the input bits are also 1:

$$\text{AND}(0,0) = 0; \quad \text{AND}(0,1) = 0; \quad \text{AND}(1,0) = 0; \quad \text{AND}(1,1) = 1$$

The OR gate is 1 when either of the input bits is 1:

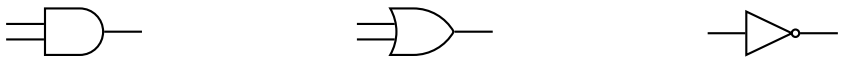
$$\text{OR}(0,0) = 0; \quad \text{OR}(0,1) = 1; \quad \text{OR}(1,0) = 1; \quad \text{OR}(1,1) = 1$$

Another important gate is the NOT gate, which simply flips the value of the input bit:

$$\text{NOT}(0) = 1; \quad \text{NOT}(1) = 0$$

We can build classical circuits for other Boolean functions $f\colon \{0,1\}^n \to \{0,1\}$ out of classical gates by creating a network of gates, where the output of a gate can be fed as input to another. Much like with quantum circuits, it can be easiest to describe a classical circuit through some sort of visualization. Pictorially, we can draw the AND, OR, and NOT gates as follows:

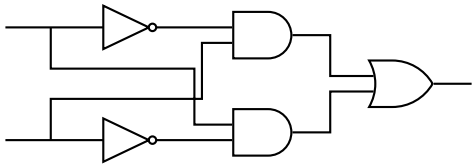
AND OR NOT



Each gate has the inputs coming in from the left and the outputs leaving from the right. To see an example of a classical circuit, let's consider the XOR function, which sums the input bits mod 2:

$$\mathrm{XOR}(0,0) = 0; \qquad \mathrm{XOR}(0,1) = 1; \qquad \mathrm{XOR}(1,0) = 1; \qquad \mathrm{XOR}(1,1) = 0$$

A classical circuit for XOR is shown below:



Notice that one key feature of the circuit above (and classical circuits in general) is that wires can be "split" or "copied" so that the value of one wire in the circuit can be fed into multiple gates.

Although it's outside the scope of this course, it will turn out that classical circuits completely capture the power of classical computation in the same way that quantum circuits completely capture the power of quantum computation.

1. Let's prove one aspect of this characterization, namely, that there are classical circuits for any Boolean function $f\colon \{0,1\}^n \to \{0,1\}$. We break this down into the following steps:

   (a) Show there is a classical circuit consisting only of 2-input AND gates that can compute that n-input AND gate. Formally, the function $\mathrm{AND}_n\colon \{0,1\}^n \to \{0,1\}$ we are trying to compute has the behavior

   $$\mathrm{AND}_n(x_1, x_2, \ldots, x_n) = \begin{cases} 1 & \text{if } x_1 = x_2 = \cdots = x_n = 1 \\ 0 & \text{otherwise} \end{cases}$$

   for all inputs $x_1, x_2, \ldots, x_n \in \{0,1\}$. You can describe your circuit for AND_n in words, or you can draw a sketch of what the circuit would look like.

   (b) Notice that we can now use AND_n gates in our circuit constructions since we've shown how to construct them from 2-input AND gates in part (a). Show that using one AND_n gate and some NOT gates, we can construct the OR_n function:

   $$\mathrm{OR}_n(x_1, x_2, \ldots, x_n) = \begin{cases} 0 & \text{if } x_1 = x_2 = \cdots = x_n = 0 \\ 1 & \text{otherwise} \end{cases}$$

   for all inputs $x_1, x_2, \ldots, x_n \in \{0,1\}$.

   (c) Suppose the function $f\colon \{0,1\}^n \to \{0,1\}$ we are trying to compute only has a single input $y = (y_1, \ldots, y_n)$ for which $f(y) = 1$. Show there is a classical circuit consisting of AND_n gates and NOT gates that computes f.

   (d) Combining all of the above, show that there is a classical circuit consisting of AND and NOT gates for *every* Boolean function $f\colon \{0,1\}^n \to \{0,1\}$.

Our goal will now be to show that for every Boolean function $f\colon \{0,1\}^n \to \{0,1\}$, there is a quantum circuit which implements the oracle O_f:

$$O_f |x\rangle |b\rangle = |x\rangle |f(x) \oplus b\rangle$$

for all $x \in \{0,1\}^n$ and $b \in \{0,1\}$. Moreover, we want this construction to be *efficient*—if there is a classical circuit which computes f with few gates, then there is also a quantum circuit which computes O_f with few gates.

2. (a) Let's first tackle the issue that classical circuits can "split" wires, whereas quantum circuits cannot. Here, we will use the fact that we are only dealing with classical bit strings, so we can copy the classical bit value into another ancilla register if we want to use it another time. Show there is a quantum circuit which implements the following "copy" operation:

$$\text{COPY} |x\rangle |0\rangle = |x\rangle |x\rangle$$

for all $x \in \{0,1\}$. What basic quantum gate implements this operation?

(b) Show there is a quantum circuit which implements the NOT operation:

$$\text{NOT} |x\rangle = |x \oplus 1\rangle$$

for all $x \in \{0,1\}$. What basic quantum gate does this operation correspond to?

(c) To implement the AND gate, we need to be more careful, since it's the first gate we've encountered which is not reversible. We will need to use some additonal registers to force the AND operation to be reversible. Show there is a quantum circuit which implements the following reversible AND operation:

$$\text{REVERSIBLE-AND} |x\rangle |y\rangle |0\rangle = |x\rangle |y\rangle |\text{AND}(x,y)\rangle$$

for all $x, y \in \{0,1\}$. What basic quantum gate implements this operation?

(d) Suppose there is a classical circuit with 2-input AND gates and NOT gates which computes some Boolean function $f\colon \{0,1\}^n \to \{0,1\}$. Combining all of the above, show there is a quantum circuit which implements the following unitary:

$$J_f |x\rangle |b\rangle |0 \cdots 0\rangle = |x\rangle |f(x) \oplus b\rangle |\text{junk}(x)\rangle$$

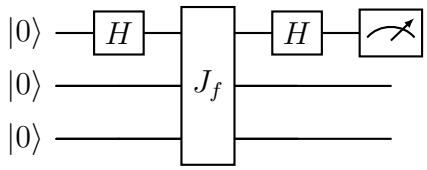
for all $x \in \{0,1\}^n$ and $b \in \{0,1\}$. Here, junk(x) is some classical bit string which depends on x.

This junk can be quite annoying if you want to implement f in superposition:

$$J_f \left(\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle |b\rangle |0 \cdots 0\rangle \right) = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle |f(x) \oplus b\rangle |\text{junk}(x)\rangle$$

The junk acts like an unwanted quantum gate. To get some intuition for how the junk can destroy the careful superposition of our state, we will look at a small example.

(e) Let $f\colon \{0,1\} \to \{0,1\}$ be the constant-0 function (i.e., $f(0) = f(1) = 0$). Consider the following circuit:



Show that the probability distribution of the partial measurement is different in the following two cases:

- junk(0) = 0 and junk(1) = 1; and
- junk(0) = 0 and junk(1) = 0.

To get around this issue we must "uncompute" the junk state, mapping it back to the all-zeros state.

(f) Notice that if we apply J_f and then $J_f^\dagger$, we will have reset the junk register, but we will have *also* reset the function register, which we don't want to do. Modify this idea slightly to obtain a complete description of a circuit that computes O_f using J_f, $J_f^\dagger$, COPY, NOT, and REVERSIBLE-AND gates. *Hint: You are allowed to introduce a new ancilla register.*

(g) Suppose there is a classical circuit with k_A 2-input AND gates and k_N NOT gates which computes some Boolean function $f\colon \{0,1\}^n \to \{0,1\}$. Write down, in terms of k_A and k_N, the number of gates needed to implement O_f in a quantum circuit. You are allowed to use the COPY, NOT, and REVERSIBLE-AND gates. Conclude by noticing that if f can be implemented efficiently by a classical circuit, then O_f can be implemented with just a constant factor overhead by a quantum circuit.