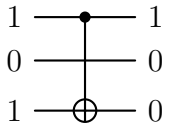
- Start as early as possible, and contact the instructor if you get stuck.
- See the course outline for details about the course's grading policy and rules on collaboration.
- Submit your completed solutions to **Crowdmark**.

1. **Self-Reducibility** [**8 marks**]
   Suppose you are given ϕ, a 3-CNF on n variables $x_1, \ldots, x_n$.
   (a) If you have an oracle that solves 3SAT, show how to construct a satisfying assignment for the variables in polynomial time.
   (b) If you have an oracle that decides whether there are more satisfying or unsatisfying assignments (ties go to satisfying assignments), then show how to use this to count how many satisfying assignments ϕ has.

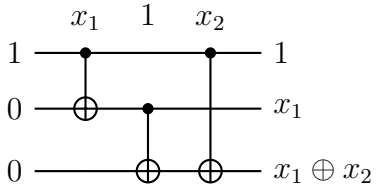
2. **Balancing formulas** [**4 marks**]
   We've seen in class that $x_1 \wedge x_2 \wedge \cdots \wedge x_n$ and $x_1 \oplus x_2 \oplus \cdots \oplus x_n$ have Boolean formulas of depth $\mathcal{O}(\log n)$. Surprisingly, *any* polynomial-size Boolean formula ϕ has an equivalent polynomial-size Boolean formula *with* $\mathcal{O}(\log n)$ *depth*.
   The proof is by divide and conquer. For sufficiently large binary trees, there exists a node u such that at most $\frac{2n}{3}$ elements are in the subtree rooted at u, and at most $\frac{2n}{3}$ elements are *outside* the subtree—an approximately equal division. For our formula ϕ, this means we can divide it into the subtree, represented by a formula $h(x_1, \ldots, x_n)$ and the context, represented by a formula $g(x_1, \ldots, x_n, u)$ which uses variable u exactly once. Thus,
   $$\phi(x_1, \ldots, x_n) = g(x_1, \ldots, x_n, h(x_1, \ldots, x_n)).$$
   To finish the inductive argument, we need to **fill in the following step**:
   Given formulas G and H equivalent[1] to g and h, construct a formula F equivalent to ϕ such that
   $$L(F) \leq k \cdot (L(G) + L(H)) + \mathcal{O}(1),$$
   $$\text{depth}(F) \leq \max\{\text{depth}(G), \text{depth}(H)\} + \mathcal{O}(1).$$
   Explain why F is equivalent to ϕ.

3. **Computation in just three bits** [**8 marks**]
   We consider a non-uniform space-limited model of computation on n-bit strings $x_1 x_2 \cdots x_n \in \{0,1\}^*$. In this model, we have three single-bit registers and a *gate* which adds bits—pick one register and add it into another one. In quantum computing, we draw such an operation like this:
   

   The horizontal lines represent the three registers, and the gate is the vertical line. The $\oplus$ of the gate is on the *target* register (the one that is added to) and the $\bullet$ is on the *control* register.

[1] Here "equivalent" means they compute the same Boolean function.

A *program* in this model is a sequence of gates labeled by an input bit x_i, the negation of an input bit, $\overline{x_i}$, or the constant 1. We *run* a program on an input $x_1 \cdots x_n$ by discarding the gates not labelled 1 (i.e., drop gates labelled x_i when $x_i = 0$ or labelled $\overline{x_i}$ if $x_i = 1$), then apply the gates to sequentially to bits 100 and accept if the last bit ends as a 1. For example, the program



computes $x_1 \oplus x_2$ in an unnecessarily convoluted way.

(a) Give a program that swaps register 1 with register 2.
(b) Give a program that computes $x_1 \wedge x_2$, and a program that computes $x_1 \vee x_2$.
(c) Prove that any polynomial-size formula F (where AND and OR have fan-in 2) can be computed by a polynomial length program.
Hint: You may apply the result claimed in the previous problem. I.e., you can assume WLOG that F is $\mathcal{O}(\log n)$ depth.

4. **Median of means** [**12 marks**] Randomization is useful not just for decision problems. It is common to have a randomized algorithm $\mathcal{A}$ which generates samples such that the mean μ is a number we want to compute. For example, suppose I sample $x, y \in [0, 1]$ uniformly at random, and output 1 if $x^2 + y^2 \leq 1$ and 0 otherwise. The mean is $\mu = \pi/4$, and the mean of many samples will approach this value. Similar strategies can be used to estimate area or volume of much more complicated shapes.

Suppose we want to estimate a number $x^* \in \mathbb{R}$, and we have an algorithm that produces a random estimate x with mean $\mu = x^*$ and known variance σ^2. This question walks through the "mean of medians" method for turning this algorithm into an estimate $\hat{x}$ with the rigorous guarantee that

$$\Pr[|\hat{x} - x^*| \leq \epsilon] \geq 1 - \delta, \tag{1}$$

or equivalently

$$\Pr[|\hat{x} - x^*| > \epsilon] \leq \delta. \tag{2}$$

(a) Suppose we use one sample x from algorithm $\mathcal{A}$. Use Chebyshev's inequality to show that we have (2) with $\delta = \mathcal{O}(\sigma^2/\varepsilon^2)$.
Reminder: Chebyshev's inequality is as follows. For any $k > 0$,

$$\Pr[|x - x^*| \geq k\sigma] \leq \frac{1}{k^2}.$$

(b) Now suppose we take n independent samples $x_1, \ldots, x_n$ from algorithm $\mathcal{A}$ and take their empirical mean $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$. Show that we can achieve (2) with any desired accuracy $\epsilon > 0$ and confidence $\delta > 0$ by taking a mean of

$$n = \mathcal{O}\Big(\frac{n}{\delta\epsilon^2}\Big)$$

samples.

(c) Now let us take $n = ms$ independent samples from $\mathcal{A}$, divided into m groups of size s. Let the estimate be $\hat{x} := \text{median}_{i\in[m]}\left(\text{mean}(x_{i1}, \ldots, x_{is})\right)$. Show that with the right balance of m and s, we can achieve accuracy $\epsilon > 0$ and confidence $\delta > 0$ with

$$n = \mathcal{O}\left(\frac{n\log(1/\delta)}{\epsilon^2}\right)$$

samples. *Hint: A Chernoff bound may be useful. For example, for independent Bernoulli*[2] *random variables* $X_1, \ldots, X_n$*, the sum* $X = \sum_{i=1}^{n} X_i$ *has the property*

$$\Pr\left[X \geq (1+a)\mathbb{E}[X]\right] \leq \exp\left(-\frac{a^2\mathbb{E}[X]}{2+a}\right) \tag{3}$$

for all $a > 0$*.*

[2] I.e., on $\{0, 1\}$