## Question 1

One method for gauging how hard it is to prove a conjecture $C$ is to investigate if NOT $C$ is true under the assumption that P equals NP. If P = NP implies NOT $C$ then proving $C$ would also prove that P $\neq$ NP, so a proof of $C$ (if true) is likely out of reach. Show that the following statements are true assuming P = NP:

(a) The problem "Given a circuit $C$ as input, all assignments are satisfying for $C$" is in P.

**Solution:** On input $C$, apply the polynomial-time algorithm for SAT to the circuit NOT $C$ and negate the answer. If all assignments to $C$ are satisfying then NOT $C$ has no satisfying assignment and the procedure accepts. Otherwise NOT $C$ has a satisfying assignment and the procedure rejects.

(b) Polynomial Identity Testing is in P. (**Hint:** Think of the randomness as a potential NP certificate.)

**Solution:** In Lecture 7 we showed that for every size-$s$ instance $C$ of polynomial identity testing that does not compute the identically zero polynomial a random assignment of the inputs from $\{1, \ldots, 3s\}$ evaluates to zero with probability at most 1/3. In particular every nonzero $C$ has at least one polynomial-size witness $x \in \{1, \ldots, 3s\}^n$ such that $C(x)$ does not evaluate to zero. Therefore the set of pairs $(C, x)$ where $C(x) \neq 0$ and $x \in \{1, \ldots, 3s\}^n$ is an NP-relation whose decision version is the complement $\overline{PIT}$ of polynomial identity testing. If P equals NP then $\overline{PIT}$ is in P and so is PIT itself as P is closed under complement.

(c) There is no polynomial-time computable family $G_n \colon \{0,1\}^n \to \{0,1\}^{n+1}$ of $(2^{n/10}, 1/4)$-pseudorandom generators. (**Hint:** The problem "On input $y$, does there exists $x$ such that $G_{|x|}(x) = y$?" is in NP.)

**Solution:** Every output of $G_n$ is a YES-instance of the problem described in the hint. The probability that a random string $Z$ in $\{0,1\}^{n+1}$ is a yes instance is at most half because only half of the strings are possible outputs of $G_n$. If P equals NP the polynomial-time algorithm $D$ for this problem is a distinguisher with advantage at least $1 - 1/2 > 1/4$. In particular this algorithm can be implemented by a family of polynomial-size circuits which fits within the required bound of $2^{n/10}$ for all sufficiently large $n$.

## Question 2

Assume $f \colon \{0,1\}^n \to \{0,1\}$ is 0.01-unpredictable against size $n^2$. Which of these constructions is an $(n^2/10, 0.1)$-pseudorandom generator? If you answer no describe a distinguisher for $G$. If you answer yes show how to convert a distinguisher for $G$ into a predictor for $f$ (possibly using results from class). Addition denotes (bitwise) xor.

(a) $G \colon \{0,1\}^{2n} \to \{0,1\}^{2n+2}$ given by $G(x, y) = (x, f(x), y, f(x) + f(y))$.

**Solution:** Yes. If not suppose $D$ has size $n^2/10$ and 0.1-distinguishes $(x, f(x), y, f(x) + f(y))$ from a random string. Let $D'$ be the circuit that takes input $(x, a, y, b)$ and applies $D$ to $(x, a, y, a + b)$. Then $D'$ has size $n^2/10 + O(1)$ and 0.1-distinguishes $(x, f(x), y, f(y))$ from a random string $(x, a, y, b)$. $D'$ must then 0.05-distinguish either of those from $(x, f(x), y, b)$. In one case by fixing $x$ (and $f(x)$) that maximizes the advantage of $D'$ we get a 0.05-distinguisher of $(y, f(y))$ from a random string of size $n^2/10 + O(1)$. In the other case by fixing $y$ and $b$ in an advantage-maximizing way we get a distinguisher of $(x, f(x))$ from random of the same advantage and size. In either case by Yao's lemma $f$ can be 0.05-predicted by size $n^2/5 + O(1)$ contradicting the assumption.

(b) $G \colon \{0,1\}^{nm} \to \{0,1\}^{\binom{m}{2}}$ (one output for every pair of inputs), with $m = 3n$, given by

$$G(x_1, \ldots, x_m) = \big(f(x_1) + f(x_2), \ldots, f(x_1) + f(x_n), f(x_2) + f(x_3), \ldots, f(x_{m-1}) + f(x_m)\big)$$

**Solution:** No. The output of $G$ includes the three bits $f(x_1) + f(x_2)$, $f(x_1) + f(x_3)$, and $f(x_2) + f(x_3)$ which always XOR to zero. The distinguisher that computes the XOR of these three bits always accepts outputs of $G$ but only accepts random strings with probability half, showing that $G$ is not even $(O(1), 1/2)$-pseudorandom.

(c) **(Optional)** $G\colon \{0,1\}^{3n} \to \{0,1\}^{3n+3}$ given by $G(x, y, z) = (x, y, z, f(x + y), f(x + z), f(y + z))$.

**Solution:** I don't know the answer to this one.

## Question 3

In Lecture 3 we showed that the following property of functions $f\colon \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$ separates *EQUALITY* (when $m = n$) from width $2^n$ read-once branching programs:

`diffext`$(f)$: For every pair $x \neq x' \in \{0,1\}^n$ there exists a $y \in \{0,1\}^m$ such that $f(x, y) \neq f(x', y)$.

(a) Argue that `diffext` is $2^{O(n+m)}$-constructive, namely describe an efficient algorithm that decides `diffext`$(f)$ using oracle access to $f$ and analyze its running time.

**Solution:** The algorithm loops over all $\binom{2^n}{2}$ pairs $x \neq x'$. For each of these pairs it tests whether any $y \in \{0,1\}^m$ violates the condition $f(x, y) \neq f(x', y)$. This condition can be checked in time $O(n+m)$, so the whole algorithm can be implemented in time $O((n + m)2^{2n+m})$. This is at most quadratic in the instance size $2^{n+m}$.

(b) Show that the probability that `diffext`$(R)$ holds for a random function $R$ is at least $1 - 2^{2n-2^m-1}$.
(**Hint:** Calculate the probability $R(x, y) = R(x', y)$ for fixed $x \neq x'$ and all $y$ and take a union bound.)

**Solution:** For fixed $x \neq x'$ the $2^{2m}$ values $R(x, y)$ and $R(x', y)$ as $y$ ranges over $\{0,1\}^m$ are uniform and independent, so the $2^m$ events $R(x, y) = R(x', y)$ are independent of probability 1/2 each. Therefore the probability that $R(x, y) = R(x', y)$ for all $y$ is exactly $2^{2^m}$. By a union bound the probability that there exist $x \neq x'$ for which this is the case is at most $\binom{2^n}{2}2^{2^m} \leq 2^{2n-1} \cdot 2^{2^m}$.

(c) Use part (b) to show that `diffext`$(f)$ is 1/2-large (and therefore natural) when $m \geq \log(2n)$.

**Solution:** When $m \geq \log(2n)$ the probability in part (b) is at least $1 - 1/2 = 1/2$ so `diffext` is 1/2-large as desired.