(1) Homer-Selman text, exercise 10.5 on page 246: prove that $\mathsf{AM} \subseteq \mathsf{NP/poly}$. Optionally, rather than the "advice" formalism, you're welcome to define $\mathsf{NP/poly}$ via polynomial-size circuits with oracle gates for a language in NP. (24 pts.)

(2) Lipton-Regan text, exercise 10.6 on page 96. Also answer whether Shor's algorithm is applicable and what its success probability would be. (30 pts. total)

(3) Given two polynomials f, g in variables $x_1, \dots, x_n$ over a field F, say that f is "Boolean equivalent" to g if every arithmetical formula ϕ for f can be transformed into a formula ψ for g by arithmetical operations and substitutions of the form $x_i^a \mapsto x_i^b$ $(a, b \geq 1)$. The substitutions do *not* have to apply to all occurrences of the variable the same way. For instance, given $\phi = (x_1 + 3x_2)(2x_1 - 6x_2x_1^2 + 1)$, you can first multiply out to get

$$2x_1^2 - 6x_1^3x_2 + x_1 + 6x_1x_2 - 18x_1^2x_2^2 + 3x_2.$$

Then you can substitute $x_1 \mapsto x_1^3$ in the fourth term to make it *cancel* the second term, and $x_1^2 \mapsto x_1$ in the first term to make it conjoin with the third, leaving

$$\psi = 3x_1 - 18x_1^2x_2^2 + 3x_2,$$

which can ultimately be reduced to the *multilinear* polynomial (formula) $3(x_1 + x_2 - 6x_1x_2)$.

You might expect that every case of equivalence should be accomplished by a short transformation that works in $O(n)$ substitutions and something like $O(n^2)$ overall bit-steps including the increased sizes of constants, but read on...

(a) Prove that the above definition of "Boolean equivalent" is equivalent to $f(x) = g(x)$ for all $x \in \{0, 1\}^n$. That is, f and g coincide on 0-1 arguments, though they might not over F in general. (It may help you to prove as a lemma that there is a unique $\hat{f}$ equivalent to f that is *multilinear*.)

(b) Show that it is NP-hard to tell whether a given formula ϕ is Boolean-equivalent to zero. Is non-equivalence NP-complete?

(c) Hence say what would follow if every formula ϕ that is Boolean-equivalent to zero has a short transformation to zero. Also explain briefly why this doesn't contradict the fact that polynomial identity testing belongs to $\mathsf{co\text{-}RP}$. ($21 + 9 + 6 = 36$ pts. total)

(4) Regarding the protocol in Chapter 12, Section 12.5.1, to prove a value

$$K = \sum_{x \in \{0,1\}^n} f(x),$$

one point that might be obscured by the inductive presentation is that the verifier V needs to *know* f in order to verify at the end that $v_n = f(r_1, r_2, \dots, r_n)$. This can become an issue when we combine with the issues in problem (3), where in particular the unique multilinear $\hat{f}$ might not be knowable to V given f.

(a) Show that if V knows $\hat{f}$, then the protocol can be restricted to work with every polynomial $p(x_i)$ communicated by the Prover being *linear*, i.e., of the form $A_ix_i + B_i$. Describe the whole protocol, following text as guide.

(b) What, then, is the probability of V catching either a wrong initial value of K or a dishonest prover when the revised protocol is run over the field $\mathbb{F}_q$ with q prime?

(c) What can you say about the complexity of the problem, given f and values $r_1, \dots, r_n \in \mathbb{F}$, of computing $\hat{f}(r_1, \dots, r_n)$? (In any event, you can conclude that having an IP protocol for this one problem would then enable you to use your proof in (a) as an alternate to the text's proof of PSPACE $\subseteq$ IP. $24 + 6 + 6 = 36$ pts.)

(5) Suppose we have a polynomial-time decidable relation $R(x, y)$ with $|y| = p = p(|x|)$ that defines a language L in NP. For any x, define $S = \{y : R(x, y)\}$ as usual. Then define the $(n+p)$-qubit quantum state

$$\Phi_x = \frac{1}{\sqrt{S}} \sum_{y \in S} e_x \otimes e_y$$

if $x \in L$, and define $\Phi_x = \frac{1}{2^{p/2}} \sum_y e_x \otimes e_y$ otherwise. Whenever $x \in L$, measuring Φ_x always gives a witness y drawn uniformly at random from S. Hence the ability to build Φ_x in random (quantum) polynomial time would put NP $\subseteq$ BQP.

Let us further suppose that given any oracle language $A \in$ BQP and relation $R^A(x, y)$ derived from a poly-time oracle NTM N^A, one can build poly-size quantum circuits C that given x construct the state

$$\Phi_x^A = \frac{1}{\sqrt{S}} \sum_{y \in S} e_x \otimes e_y,$$

where $S = \{\, y : R^A(x, y) \,\}$ if $x \in L(N^A)$ and $S = \{\, 0, 1 \,\}^p$ otherwise.

(a) Show that *then* the entire polynomial hierarchy would be inside BQP.

(b) Make the argument of (a) work even if we can only build a less-helpful quantum state Ψ^A such that when we measure the ancilla qubit i, we get 1 with probability $p_i = \Pr_{y \in S}[y_i = 1]$. (*Hint:* When y is *unique* then Ψ_x^A gives bit y_i with certainty and so is just as helpful as Φ_x^A. When y is not unique then even measuring all i might not give a witness: if $S = \{\, 0^p, 1^p \,\}$ the result is the same "random noise" $p_i = 0.5$ as when $x \notin L(N^A)$.)

(c) For a "side question," can you give the notation for the state Ψ^A? It is a non-entangled state and so has a tensor product in place of the $\sum_y$ summation. ($18 + 18 + 3 = 39$ pts., for 165 total)

Footnote: It is not known whether $\mathsf{BQP}^{\mathsf{BQP}} = \mathsf{BQP}$ in any kind of analogy to $\mathsf{BPP}^{\mathsf{BPP}} = \mathsf{BPP}$. Evidently because of this, it seems not known whether NP $\subseteq$ BQP alone suffices to collapse the polynomial hierarchy inside BQP. Nor is BQP $\subseteq$ PH known the way we saw BPP $\subseteq \Sigma_2^p \cap \Pi_2^p$ in the course; I have an idea for BQP $\subseteq \Sigma_3^p \cap \Pi_3^p$ but there are oracles D relative to which BQP^D is not PH^D at all. One can abstract the above handling of "Φ_x" into a notion of "BPQP" where a non-quantum BPP-machine (or really a ZPP-machine) builds with high probability a quantum circuit which is then evaluated to build a state whose measurement gives a witness with high probability. The idea is that unlike when you relativize BQP^C, the oracle machine uses only classical randomness to make queries and certainly can't superpose its *queries* like with Deutsch's Algorithm. I *still* couldn't show $\mathsf{BPQP}^{\mathsf{BPQP}} = \mathsf{BPQP}$, however. If you can get that or some other way to make the mechanism of Toda's Theorem work without "piggybacking" the oracle A as above, then this is open-ended large extra-credit.

Despite the length and variety of the elements in this problem, the answers follow patterns of lectures and earlier HW and don't have to be much longer than this footnote. At least it's one question that combines most of the parts of the course. Happy solving. . .