Answer to Question 1:
Justifications are presented for educational purposes only. They were not required for full credit.
a) True. A checker is just a particular type of oracle proof system.
b) Unknown

c) True. If $L$ is in $\text{BPP} / \text{poly}$, it is accepted by a probabilistic polynomial-time machine $M$ that takes polynomial-length advice $\{\alpha_n\}_{n \geq 0}$. $M$ can be simulated by a deterministic polynomial-time machine $M'$ that takes polynomial-length advice $\{\beta_n\}_{n \geq 0}$. So $M'$ with advice $\{\alpha_n \cdot \beta_n\}_{n \geq 0}$ accepts $L$.
d) False. $\text{BPP}$ is contained in $\Sigma_P^2 \cap \Pi_P^2$ (the Sipser-Gacs Theorem), but $\text{BPP} / \text{poly} = \text{P} / \text{poly}$ contains undecidable languages, and $\Sigma_P^2 \cap \Pi_P^2$ does not.
e) Unknown

Answer to Question 2:
a) See Definition 8.26. Basically, a checker for $L$ is an oracle proof system for $L$ (as in MIP) in which the oracle that causes the base machine to accept all $x \in L$ is an $L$ oracle. In an ordinary one-prover interactive proof system for $L$, the correct prover that causes the verifier to accept whenever $x \in L$ is not restricted to the power conferred by an $L$ oracle. Furthermore, the “cheating” prover that tries to make the verifier accept when $x \not\in L$ can be adaptive; by contrast, in Definition 8.26, the program that is incorrect on input $x$ (when the checker is supposed to reject) is not adaptive.
b) As explained in Section 8.6.1, the interactive proof system for TQBF is a checker, because the prover can be implemented using a TQBF oracle.\textsuperscript{1} By contrast, in the interactive proof system for coSAT, we do not know how to do the algebraic computations required of the prover simply by making SAT queries.

\textsuperscript{1}In fact, the prover in any interactive proof system can be implemented using a TQBF oracle, because TQBF is \text{PSPACE}-complete, and we showed in HW exercise 8.1 that optimal prover answers can be computed in polynomial space.
Answer to Question 3:

a) \[ a : T \mapsto 1 \]
\[ a : F \mapsto 0 \]
\[ a : x_i \mapsto X_i \]
\[ a : \neg x_i \mapsto (1 - X_i) \]
\[ a : \phi_1 \lor \phi_2 \mapsto a(\phi_1) + a(\phi_2) - a(\phi_1) \cdot a(\phi_2) \]
\[ a : \phi_1 \land \phi_2 \mapsto a(\phi_1) \cdot a(\phi_2) \]

b) The degree of \( h_1 \) is \( O(m) \), and the degree of \( h_2 \) is \( O(m \cdot 2^{\lceil n/2 \rceil}) \).

c) See Formula 8.13 and the discussion immediately following it. The linearization operator is used for degree reduction.

Answer to Question 4:

a) Using formula (17.6), we can add 1 to the output of algorithm \( B \) to flip the parity of the number of satisfying assignments. That is,
\[ \phi \in \text{coSAT} \rightarrow \text{Prob}[(B(\phi) + 1) \in \oplus SAT] \geq 1 - 2^{-m} \]
\[ \phi \notin \text{coSAT} \rightarrow \text{Prob}[(B(\phi) + 1) \in \oplus SAT] \leq 2^{-m} \]

b) See Lemma 17.21. The reduction is oblivious in the sense that the formula \( \tau \) depends only on the number \( n \) of inputs to the function \( \beta \) and not on the function itself.

c) An instance \( \phi \) of \( \Sigma_c^{\text{SAT}} \) is of the form
\[ \phi(x_1, x_2, \ldots, x_c) = \exists x_1 \forall x_2 \cdots Q_c x_c \phi'(x_1, x_2, \ldots, x_c) \]
where each of the \( x_i \)'s is a string of boolean variables, and \( Q_c \) is \( \exists \) if \( c \) is odd and \( \forall \) if \( c \) is even. Note that \( \phi \) is of the form \( \exists x_1 \psi(x_1) \), where \( \psi(x_1) \) is an instance of \( \Pi_{c-1}^{\text{SAT}} \). By our inductive hypothesis, for any \( m \in N \), there is a probabilistic, polynomial-time algorithm \( f \) such that, for any \( x_1 \), with probability at least \( 1 - 2^{-m} \), \( \psi(x_1) \in \Pi_{c-1}^{\text{SAT}} \) if and only if \( \rho(z, x_1) \in \oplus \text{SAT} \), where \( \rho(z, x_1) = (f(\psi(x_1)))(z) \).

Our desired function \( \beta(x_1) \) is \( \oplus \rho(z, x_1) \). That is, \( \beta(x_1) = 1 \) if and only if the number of \( z \) that satisfy \( \rho(z, x_1) \) is odd.

Answer to Question 5:

a) See Definition 8.10 and the paragraph immediately following. In particular, \( \text{AM} \) is the class of languages with interactive proof systems in which the verifier sends a random string \( r \), the prover responds with a message \( m \), and the verifier’s decision is obtained by applying a deterministic polynomial-time function to \( r, m \), and the input \( x \). If \( x \) is (resp., is not) in the language, then the verifier accepts (resp., rejects) with probability at least \( \frac{2}{3} \).

b) We saw in HW exercise 8.3 that \( \text{AM}[2] = \text{BP} \cdot \text{NP} \), which is defined as \( \{ L : L \leq_r 3\text{SAT} \} \).

Note that \( \text{BP} \cdot \text{NP} \subseteq \text{BPP}^{\text{NP}} \). (Because \( \text{BPP}^{\text{NP}} \) contains \( \text{coNP} \), this inclusion is believed to be proper – see HW exercise 7.8.) By the Sipser-Gacs Theorem, \( \text{BPP} \subseteq \Sigma_2^P \), and the proof of Sipser-Gacs relativizes. Therefore, \( \text{BP} \cdot \text{NP} \subseteq \text{BPP}^{\text{NP}} \subseteq (\Sigma_2^P)^{\text{NP}} = \Sigma_3^P \).