Instructions: Submit one writeup per group; please discuss within your group – other resources including the Web are not allowed for consultation. Write your solutions neatly; write only your group ID, not your names. For all problems, we assume that we have fixed the alphabet Σ = \{0, 1\}, and hence that all languages we are referring to are subsets of \{0, 1\}^*.

1. Let pos-or-neg-3sat denote the class of satisfiable 3-SAT formulas in which each clause has all literals as positive variables, or has all literals as negative variables. Prove that pos-or-neg-3sat is NP-complete. (10 points)

2. Let c be some arbitrary (constant) positive integer. Let clique-c be the clique problem restricted to graphs in which each vertex has at most c neighbors.
   (a) Is clique-c in NP? Justify your answer. (5 points)
   (b) Prove that if clique-c is NP-complete for any particular c, then P = NP. (10 points)

3. Suppose, as usual, that we have fixed the alphabet Σ = \{0, 1\}, and that all languages we are referring to are subsets of \{0, 1\}^*.
   (a) Suppose P = NP. Then show that all but very few languages in NP are NP-complete. What are these “very few” exceptional languages? (10 points)
   (b) Consider the following conjecture C: “For any two NP-complete languages L_1 and L_2, there is a function \( f : \{0, 1\}^* \rightarrow \{0, 1\}^* \) such that: (i) \( f \) is one-to-one and onto; (ii) given \( x, f(x) \) and \( f^{-1}(x) \) can be computed in polynomial time, and (iii) for all \( x, x \in L_1 \) if and only if \( f(x) \in L_2 \).” Suppose C is true; then show that \( P \neq NP \). (Use part(a) if necessary.) (10 points)

4. Consider the recurrence where \( T(1) \) is some constant, and for some constant \( c > 0 \), \( T(n) \leq 2T(n-1) + n^c \) for \( n \geq 2 \). Is there some finite function \( f(c) \) such that \( T(n) \leq f(c) \cdot 2^n \)? Justify your answer. (10 points)

5. Let us develop a nondeterministic polynomial-time algorithm to certify if a given number \( N \) is a prime. (Note that “polynomial time” here refers to some polynomial of \( \log N \), not a polynomial of \( N \).) That is, develop a prover-verifier pair where the prover can make nondeterministic guesses and can send one polynomial-length message to the verifier; the verifier runs in deterministic polynomial time and accepts the prover’s “proof” iff \( N \) is prime. You may use the following fact: if \( N \geq 3 \), then \( N \) is a prime iff there exists an integer \( a \), \( 2 \leq a \leq N - 1 \), such that:
   - \( a^{N-1} \), when divided by \( N \), leaves a remainder that is equal to 1, and
   - for all prime factors \( p \) of \( N - 1 \), \( a^{(N-1)/p} \), when divided by \( N \), leaves a remainder that is not equal to 1.
   (a) Carefully explain a (recursive) nondeterministic prover-verifier protocol as above, for certifying that \( N \) is prime. (5 points)
   (b) Write a recurrence relation for the time complexity of your protocol, and analyze it rigorously to show that your protocol runs in nondeterministic polynomial time. (10 points)