Problem 1

Let \( CNF_2 = \{ \langle \phi \rangle \mid \phi \text{ is a satisfiable cnf-formula where each variable appears in at most } 2 \text{ places} \} \). Show that \( CNF_2 \in P \).

Problem 2

Let \( SATRange \) be a language defined as follows:

\[
SATRange = \{ \langle \phi, lo, hi \rangle \mid hi \geq lo \geq 0, \phi \text{ is a Boolean formula with } lo \leq k \leq hi \text{ satisfying assignments, } lo \text{ and } hi \text{ are integers represented in binary} \}
\]

1. Show that \( SATRange \) is in PSPACE.
2. Show that \( SATRange \) is NP-hard.
3. Show that \( SATRange \) is coNP-hard.
4. Show that if \( SATRange \) is in NP, then NP=coNP.
5. Give a polynomial-time algorithm that, on input a Boolean formula \( \phi \), determines the number of satisfying assignments for \( \phi \) using an oracle for the language \( SATRange \).
Problem 3

Consider the following two decision problems:

**SubsetSum:**

- **Input:** A set \( Q = \{a_1, a_2, \ldots, a_n\} \) of positive integers and a positive integer \( d \).
- **Output:** “Yes” if there is a subset of \( Q \) that adds to \( d \).
- **Example:** \( Q = \{8, 15, 12, 5, 2, 4\}, \ d = 13 \) is a “Yes” instance of SubsetSum since \( \{8, 5\} \subseteq Q \) and \( 8 + 5 = 13 \).

**Packing:**

- **Input:** A set \( S \) of \( n \) rectangles of varying size \( S = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\} \), and a single rectangle with dimensions \((d_x, d_y)\).
- **Output:** “Yes” if there is some way of fitting a subset of the boxes inside the target dimensions so that all of the space is filled.

We wish to show that Packing is NP-hard, given that SubsetSum is NP-complete.

1. Consider the following proof:

   Given an input \((Q, d)\) to the SubsetSum problem, we construct an instance of the Packing problem by taking each element \( x \) of \( Q \) and adding a box of size \((x, x)\) to the input \( S \) for the Packing problem. Then use the target \( d \) to construct the target dimensions \((d, d)\), and pass that problem off to Packing.

   This is clearly a polynomial-time reduction, because it requires going through all of the elements of the input in linear time. In order to show that it is correct, we must show that an instance of the SubsetSum problem can be solved if and only if the corresponding Packing problem has a solution.

   One way to do this is to show that a solution to the SubsetSum problem can be transformed into a solution to the corresponding Packing problem. We can do so by choosing the boxes corresponding to the
numbers in the correct subset, and placing them in the target rectangle. Because the sides of the chosen rectangles sum to the target value \( d \), the boxes completely fill the \( d \times d \) space.

Therefore, we have a reduction from SUBSETSUM to PACKING, and so PACKING is at least as hard as SUBSETSUM, and must therefore be NP-hard.

Explain why the above proof is wrong. Give a counterexample.

2. Provide a correct proof.