Revision Questions

1. **Does 2-SAT, the complement of 2-SAT, belong to P?**

- 2-SAT is a special case of the Boolean satisfiability problem where each clause has exactly 2 literals. It's known that 2-SAT is in P and can be solved in polynomial time.

- The complement of 2-SAT is also in P because if a problem is in P, its complement is also in P. This is a property known as closure under complement.

2. **Does 3-SAT, the complement of 3-SAT, belong to NP?**

- 3-SAT is the classic NP-complete problem, which means it's NP-hard and in NP. However, the complement of an NP-complete problem is not necessarily in NP. It could be in co-NP, but we don't know for sure.

- Whether 3-SAT's complement is in NP or co-NP is an open question and is part of the famous P vs. NP problem, which remains unresolved.

3. **Can you define co-NP, by analogy with co-r.e.?**

- Co-NP is the class of decision problems for which the answer is "no." In other words, a language L is in co-NP if and only if its complement (the set of strings not in L) is in NP.

- This is similar to the definition of co-recursively enumerable (co-r.e.) in the context of recursive enumerability. If a problem is in co-NP, it means that a proposed solution can be efficiently verified.

4. **What is the class of languages that have a verifier (with no requirement for it to be polynomial time)?**

- The class of languages for which there exists a verifier (a machine that can efficiently check the correctness of a proposed solution) without the requirement for it to be polynomial time is known as the class **RE** (Recursive Enumerable).

- RE includes all problems for which a proposed solution can be checked by a Turing machine, even if the verification process may not terminate for some inputs. These problems may not be decidable (i.e., they may not always halt), but they are at least recursively enumerable.