Homework 12: Due Wednesday, April 15

Problem 1:
a. Write a Scheme function total-num-connectives that takes an input, assumed to be a propositional formula, and outputs the total number of connectives in the formula (where you count with repetition, so 3 different ANDs would contribute 3 to the sum).
b. Write a Scheme function total-num-symbols that takes an input φ, assumed to be a propositional formula, and outputs the total number of propositional symbols in φ (where you count with repetition, so 4 occurrences of P would contribute 4 to the sum).

Problem 2:
a. Prove the following by structural induction on propositional formulas: If φ is a propositional formula with n total connectives, then φ has at most $n+1$ many total propositional symbols (where we are interpreting *total* as in Problem 1).
b. Give an example of a propositional formula, with some total number n of connectives, that has strictly less that $n+1$ many total propositional symbols.

Problem 3: Write a Scheme function satisfiable? that takes an input φ, assumed to be a propositional formula, and outputs a boolean according to whether there exists at least one truth assignment making φ true.

Problem 4: We say that two propositional formulas φ and ψ are *equivalent* if they evaluate to the same outcomes for all truth assignments (defined on the variables that occur in at least one of φ and ψ). For example, the formulas

$$\mathsf{P} \wedge (\mathsf{Q} \vee \mathsf{R}) \qquad \text{and} \qquad (\mathsf{P} \wedge \mathsf{Q}) \vee (\mathsf{P} \wedge \mathsf{R})$$

are equivalent, which you can check by examining all 8 truth assignments (or by letting Scheme output the truth tables). By considering all four possibilities for the truth values of φ and ψ, notice that $(\varphi \to \psi)$ is equivalent to $((\neg\varphi) \vee \psi)$ for all formulas φ and ψ.
a. Write a Scheme function that takes an input, assumed to be a propositional formula, and outputs an equivalent propositional formula that has no IMPLIESs.
b. Write a Scheme function that takes an input, assumed to be a propositional formula, and outputs an equivalent propositional formula with no IMPLIESs, and such that the only NOTs are applied directly to propositional symbols. In other words, it is fine to have (NOT P) inside your formula, but (NOT (AND P Q)) is not permitted. Explain the theory behind why your program works.