1. Show that logspace computable functions are closed under composition. So if \( f \) and \( g \) are each computable in logspace, then so is their composition.

*Note:* Given \( x \) you need to compute \( g(f(x)) \) in logspace. You don’t have enough space to write down \( f(x) \), which is potentially long.

*Hint:* Show how to do the following: given \( i \) (number written on a work tape) and \( x \) (on the read-only input tape), show how to compute the \( i \)th bit of \( f(x) \) in logspace. Now think of the logspace computation that takes \( f(x) \) as input and computes \( g(f(x)) \). Combine these two computations together.

2. \( \#P \)-completeness:

   (a) Show that the following problem is \( \#P \)-complete:

   \( \#\text{DNF-SAT} \): Given a boolean formula \( \phi \) in DNF (or-of-ands-of-literals), determine how many satisfying assignments \( \phi \) has.

   (b) Show a parsimonious reduction from \( \#\text{DNF-SAT} \) to the following problem:

   \( \#\text{REGEX} \): Given a regular expression \( \alpha \) over \( \{0, 1\} \), and a string \( x \), determine how many strings of length \( |x| \) are accepted by \( \alpha \). Note that the argument \( x \) is just used as a unary representation of a number (similar to how we defined the canonical NP-complete problem).

3. Let \( f \) and \( g \) be two functions with integer outputs and let \( \alpha > 1 \) be a real number. We say that \( g \) is an \( \alpha \)-approximation of \( f \) if \( f(x)/\alpha \leq g(x) \leq \alpha f(x) \) for all inputs \( x \).

   (a) Given a boolean formula \( \phi \) let \( \#\phi \) denote the number of satisfying assignments. Given \( \phi \), describe how to construct a new formula \( \phi' \) where \( \#\phi' = (\#\phi)^2 \).

   (b) Suppose there exists a polynomial-time algorithm that is a 2-approximation to the \( \#\text{SAT} \) problem. Show that there also exists a polynomial-time \( \sqrt{2} \)-approximation algorithm to the \( \#\text{SAT} \) problem.

   (c) Suppose there exists a polynomial-time algorithm that is a 2-approximation to the \( \#\text{SAT} \) problem. Show that \( P = NP \).

   *Hint:* How close would the approximation ratio \( \alpha \) have to be to 1 in order for an \( \alpha \)-approximation to actually be an exact algorithm for \( \#\text{SAT} \)? Can you extend part (b) to drive \( \alpha \) low enough?