**Problem 1**
Recall that a function \( f : \mathbb{N} \rightarrow \mathbb{N} \) is said to be computable if there is a Turing machine \( M_f \) such that on input \( k \), \( M_f \) halts with \( f(k) \) written on its tape. For a function \( f : \mathbb{N} \rightarrow \mathbb{N} \) define \( \text{Range}(f) = \{ f(k) \mid k \in \mathbb{N} \} \).

Let \( f \) be a 1-to-1, computable function such that \( \text{Range}(f) \) is recursive. (Note, in general, computability of \( f \) only guarantees that \( \text{Range}(f) \) is recursively enumerable.) Let \( g \) be a computable function that dominates \( f \), i.e., there is \( n_0 \) such that for all \( n > n_0 \), \( f(n) \leq g(n) \). Prove that \( \text{Range}(g) \) is also recursive.

**Problem 2**
Given a CFG \( G \) over \( \{a, b\} \), show that checking if there is a string with equal number of \( a \)s and \( b \)s in \( L(G) \) is decidable. You may want to use the following result due to Rohit Parikh, that is described below.

The Parikh image of a string \( w \in \Sigma^* \) is a function \( \psi_w : \Sigma \rightarrow \mathbb{N} \) that maps each symbol \( a \in \Sigma \) to the number of times it appears in \( w \). For example, for \( w = aaababab \) we have \( \psi_w : \{a, b\} \rightarrow \mathbb{N} \) defined by \( \psi_w(a) = 5 \) and \( \psi_w(b) = 3 \). The Parikh image of a language \( L \) is the set of functions \( \psi_L = \{ \psi_w \mid w \in L \} \).

**Parikh's Theorem:** Given a CFG \( G \), there is an algorithm that constructs a DFA \( A \), such that \( \psi_{L(G)} = \psi_{L(A)} \).

**Problem 3**
Consider "limit-TMs" which are TMs with a read-write input tape, and a write-only output tape, such that every transition specifies an output of either 0 or 1 on the output tape. Thus, every computation of \( M \) on some input word \( w \) necessarily produces an infinite binary sequence which we denote by \( M(w) \).

If the sequence \( M(w) \) converges to 1, we say that \( M \) accepts \( w \) in the limit. If \( M(w) \) converges to 0, we say that \( M \) rejects \( w \) in the limit. A language is limit-acceptable iff there is a TM that accepts each \( w \in L \) in the limit. A language is limit-decidable iff it is limit-acceptable and furthermore, for each \( w \notin L \), \( M \) rejects \( w \) in the limit.

1. Is the diagonal language \( L_d = \{ i : M_i \text{ does not accept } i \} \) limit-decidable, limit-acceptable but not limit-decidable, or neither limit-acceptable nor limit-decidable? Prove your answer correct.
2. Give a language that is limit-acceptable but not limit-decidable. Prove your answer correct.

**Problem 4**

Recall that satisfiability of CNF formulas, SAT is NP-complete, and the corresponding counting problem #SAT is #P-complete. Here we consider a related language DNFSAT and #DNFSAT. A DNF formula consists of a disjunction (OR) of several terms, where each term is a conjunction (AND) of several literals. A DNF formula \( \phi \) belongs to DNFSAT if there is an assignment to its variables such that \( \phi \) is satisfiable. #DNFSAT is the problem of computing the number of satisfying assignments to a given DNF formula.

1. What is the lowest complexity class you can place DNFSAT in? (Consider only complexity classes L, NL, P, NP, PSPACE and their co-classes.)

2. Show that #DNFSAT is #P-complete (w.r.t polynomial time Turing reductions).

3. For sets \( S_1, \ldots, S_m \subseteq \{0,1\}^n \), suppose for each \( i = 1, \ldots, m \), \( a_i := |S_i| \) is given, and also an oracle is given which, for any \( i \in [m] \) and any \( J \subseteq [m] \) gives an approximation of \( p_{i,J} := \Pr_{x \in \{0,1\}^n}[x \in \bigcup_{j \in J} S_j | x \in S_i] \). More precisely, on input \((i, J, \epsilon)\) for \( i \in [m] \), \( J \subseteq [m] \) and \( 1/\epsilon \) a polynomial, the oracle returns \( b(i, J, \epsilon) \in [p_{i,J} - \epsilon(m,n), p_{i,J} + \epsilon(m,n)] \).

   Give a polynomial time algorithm that, given a polynomial \( p \), outputs a number in the range \( |\bigcup_{i=1}^m S_i| (1 \pm 1/p(mn)) \).

4. Show that #DNFSAT can approximated within a \((1 + 1/poly)\) multiplicative factor, with (say) probability \( \frac{2}{3} \), in polynomial time. (Polynomials in the size of the formula.) You can use the result from the previous sub-problem.