1. A palindrome is a word \( w_1 w_2 \ldots w_k \) whose reverse \( w_k \ldots w_1 \) is the same string (e.g., danaranad). Consider a string \( A = a_1 a_2 \ldots a_n \). A partitioning of a string is a palindrome partitioning if every substring of the partition is a palindrome. For example, \( aba|b|bbabb|aba \) is a palindrome partitioning of \( ababbbabbaba \). Design a dynamic programming algorithm to determine the coarsest (i.e., fewest cuts) palindrome partitioning of \( A \).

   a) Formally define the set of subproblems you will solve.

   b) Give your recurrence for the solution of a given subproblem in terms of other subproblems.

   c) Give a non-recursive pseudo-code specification of the algorithm and state its complexity in terms of \( n \).

2. Double-SAT is a problem for which you are given a boolean formula \( \Phi \) that is a conjunction of disjunctions (just like SAT). An algorithm for Double-SAT should answer YES if there are at least two satisfying assignments to \( \Phi \) and should answer NO if there is only one or none.

   Prove that Double-SAT is NP-Complete.

3. What is the expected number of collisions when using a random hash function from a 2-universal family to hash \( n \) elements of a universe \( M \) into a table of size \( 2n \)?

4. We are given two strings \( x \) and \( y \) of length \( m \) and \( n \) respectively. We are asked to find the new edit distance between these two strings. That is, the minimum number of operations needed to transform \( x \) to \( y \) when these types of operations are allowed: (1) insert a character in any position, (ii) change one character into another, (iii) delete a whole consecutive block of characters of \( x \). Each of these three operations counts as one step. Find a dynamic programming algorithm that solves this problem, as follows: Define, for \( i = 0, \ldots, m \) and \( j = 0, \ldots, n \), \( ED[i, j] \) to be the edit distance between the first \( i \) characters of \( x \) and the first \( j \) characters of \( y \).

   (Extra credit) Can you devise an \( O(m \cdot n) \) algorithm for this problem?