1. **Making change with strange coins.** Consider the problem of making change for \( n \) cents out of the fewest number of coins. You previously designed a greedy algorithm that gave optimal solutions for US and European coins, but also found counterexamples where the greedy algorithm doesn’t give an optimal solutions.

(a) Design a dynamic programming algorithm that takes as input the amount of change to make, \( n \), and a list of coin denominations, \( c_1, \ldots, c_k \). If it is possible to make \( n \) cents in change using denominations \( c_1, \ldots, c_k \), your algorithm should output the smallest number of coins required. Otherwise, your algorithm should output FAIL.

(b) How would you need to modify your algorithm to also output the coins used to make optimal change?

2. **Polynomial-time Verifiers.** Call \( V \) an efficient verifier for a decision problem \( L \) if

- \( V \) is a polynomial-time algorithm that takes two inputs \( x \) and \( w \).
- There is a polynomial function \( p \) such that for all strings \( x, x \in L \) if and only if there exists a certificate string \( w \) such that \( |w| \leq p(|x|) \) and \( V(x, w) = \text{yes} \).

The following problems are not known to have polynomial-time algorithms. For each problem describe a polynomial-time verifier (and the corresponding certificates).

(a) **THREE-COLORING.** Given \( G = (V, E) \) return yes iff the vertices in \( G \) can be colored using at most three colors such that for every edge \( e = (u, v) \) in \( E \), \( u \) and \( v \) have different colors.

(b) **WEDDING-PLANNER.** Recall in the Wedding planner problem, the input consists of a list of \( n \) people to possibly invite to a wedding, along with \( m \) clauses, where each clause specifies some criteria for whom to invite or not invite. Output yes iff there exists an invitation list that satisfies all clauses.

   **Note:** Assume that each clause is of the form e.g. \( x_1 \lor \bar{x}_2 \lor \cdots \lor \bar{x}_k \), where \( x_i \) means to invite person \( i \), and \( \bar{x}_j \) means to not invite person \( x_j \).

(c) **INDEPENDENT-SET** Given an undirected graph \( G = (V, E) \) and integer \( k \) and returns yes iff \( G \) contains an independent set of size at least \( k \). An independent set is a set of vertices with no edges between them: \( W \subseteq V \), and \( u, v \in W \Rightarrow (u, v) \notin E \).

(d) **VERTEX-COVER** takes an undirected graph \( G = (V, E) \) and integer \( k \) and returns yes iff \( G \) contains a vertex cover of size at most \( k \). A vertex cover is a set of vertices such that every edge has at least one end in the set: \( W \subseteq V \), and \( \forall (u, v) \in E, u \in W \) or \( v \in W \).

(e) **FACTORIZING.** Given numbers \( n, k \) written in binary, output yes iff \( n \) is divisible by \( d \) for some \( 1 < d \leq k \).

(f) **NOT-FACTORIZING.** Given numbers \( n, k \) written in binary, output yes iff \( n \) is NOT divisible by \( d \) for any \( 1 < d \leq k \). **Hint:** The problem PRIMES is solvable in polynomial time.

**PRIMES:** Given a number \( n \) written in binary, output yes iff \( n \) is a prime number.