1. Recall the coin changing problem: Given denominations \( d = (d_1, d_2, \ldots, d_n) \) and an amount \( N \), determine the number of coins in each denomination necessary to disburse \( N \) units using the fewest possible coins. Assume that there is an unlimited supply of coins in each denomination.

   a. Write pseudo-code for a greedy algorithm which attempts to solve this problem. (Recall that the greedy strategy doesn't necessarily produce an optimal solution to this problem. Whether it does or not depends on the denomination set \( d \).) Your algorithm will take the array \( d \) as input and return an array \( g \) as output, where \( g[i] \) is the number of coins of type \( i \) which are to be disbursed. Assume that the denominations are indexed by increasing value \( d_1 < d_2 < \cdots < d_n \), so that your algorithm will step through array \( d \) in reverse order. You may also assume that \( d_1 = 1 \) so that it is possible to pay any amount.

   b. Suppose \( d_i = b^{i-1} \) for some integer \( b > 1 \), and \( 1 \leq i \leq n \), i.e. \( d = (1, b, b^2, \ldots, b^{n-1}) \). Does the greedy strategy always produce an optimal solution in this case? Either prove that it does, or give a counter-example.

   c. Suppose \( d = (1, 5, 10, 25) \). Prove that the greedy strategy produces an optimal solution for any amount \( N \).

   d. Suppose that \( d_1 = 1 \) and \( 2d_i \leq d_{i+1} \) for \( 1 \leq i \leq n - 1 \). Does the greedy strategy always produce an optimal solution in this case? Either prove that it does, or give a counter-example.

2. **Bar Weighing Problem:** Assume we are given 12 gold bars numbered 1 to 12 where 11 bars are pure gold and one is counterfeit: either gold-plated lead (which is heavier than gold), or gold-plated tin (lighter than gold). The problem is to find the counterfeit bar and what metal it is made of using only a balance scale. Any number of bars can be placed on each side of the scale, and each use of the scale produces one of three outcomes, indicating either that both sides are the same weight, or which of the two sides is heavier.

   a. Give a decision tree argument to establish a lower bound on the (worst case) number of weighings which must be performed by any algorithm which solves this problem.

   b. Design an algorithm which solves this problem with the fewest possible (worst case) number of weighings. Represent your algorithm as a decision tree. (You may also write it in pseudo-code, though this is not required.)