Please respect the following guidelines for writing pseudocode:

1. C instructions are fine. But do not write object-oriented additions. Do not declare or use any class. Declare only procedures (if necessary) and explain in words what each procedure does, and what is the use of each parameter.

2. One instruction per line

3. Match the brackets with a horizontal line

4. Number your lines

5. Write down if your array is indexed 0…n − 1 or 1…n

**Problem 1** Describe an efficient greedy algorithm for making change for a specified value using a minimum number of coins, assuming there are four denominations of coins (called quarters, dimes, nickels, and pennies), with values 25, 10, 5, and 1, respectively. Argue why your algorithm is correct.

**Problem 2** Give an example set of denominations of coins so that a greedy change making algorithm will not use the minimum number of coins. Give an instance, show the output of the greedy algorithm on this instance, and show better output.

**Problem 3** What is the best way to multiply a chain of matrices with dimensions that are 10×5, 5×2, 2×20, 20×12, 12×4, and 4×60? Show your work.

**Problem 4** Suppose we are given a collection \( A = a_1, a_2, \ldots, a_n \) of \( n \) positive integers that add up to \( N \). Design an \( O(nN) \)-time algorithm for determining whether there is a subset \( B \subseteq \{1, 2, \ldots, n\} \), such that \( \sum_{i \in B} a_i = \sum_{i \in \{1, 2, \ldots, n\} \setminus B} a_i \).

*Hint:* Use dynamic programming.

**Problem 5** Given an array \( A \) with \( n \) integers, a monotonically increasing subsequence is a sequence of (not necessarily consecutive) indexes \( i_1 < i_2 < \ldots < i_k \) such that \( A[i_1] < A[i_2] < \ldots < A[i_k] \); the length of subsequence is \( k \). Given an \( O(n^2) \) algorithm to find a longest monotonically increasing subsequence. Present pseudocode and analyze the running time.

*Hint:* Dynamic programming.