Problem 1  (20 points)

You are given \( n \) dogs of different weights (not necessarily distinct) ascendingly sorted by their ages (distinct). The dogs have their weights (between 1 & \( M \) pounds) and Ages (in days) expressed in integers. They need to be transported to a dog show in a trailer having a lot of compartments (\( C_1, C_2, \ldots \)). The rules of putting the dogs in different compartments are:

- Every compartment (except the last) should carry a weight of exactly \( M \) pounds, so that the connecting cable between two adjoining compartments is not stressed.
- No dog travels in a lower-indexed compartment than an younger one (i.e, if a dog is traveling in \( C_5 \), all dogs younger than him must travel in \( C_1 \) through \( C_5 \) too).
- Once the dogs are loaded, every compartment must also be loaded with a single cotton bag to make up for the unused weight. However, cotton bag’s cost increases quadratically with its weight.

Design an efficient loading algorithm to minimize the cost of the cotton bags.

Problem 2  (10 points)

There is a strange species of fish where a parent fish would eat up its own immediate offspring fishes if they come in contact. Now, you are given a single big family of fishes of different weights (including the greatest ancestor of the family). Your job is to design an algorithm to select a bunch of fishes for your aquarium with the constraint that none of them would eat each other and their total weight is maximized. You can assume a tree model for representing the parent-child relationship. Derive the time complexity of your algorithm.

As for example, here the possible grouping of fishes who can peacefully co-exist are (A, D, E, F, G, H) or (B, C) or (B, F, G, H) or (C, D, E) or any of their subsets.

Problem 3  (15 points)

There is a group of people \( P \) (of size \( n \)) standing in a queue in front of a TV. As is the common experience, if any of the persons standing in front of a person \( A \) is taller than \( A \), then \( A \) cannot see the TV. Design a Dynamic-programming algorithm to find a maximum-sized group of people (say \( G \)) from \( P \) so that the relative standing order of the people in \( G \) is exactly the same as in \( P \). That is, if person \( i \) is standing somewhere behind person \( j \) in \( G \), \( i \) would be standing somewhere behind \( j \) in \( P \) too. You can assume all the heights to be integral and distinct.
Problem 4. (10 points)

Let \( A = \{A_1, \ldots, A_n\} \) be a set of distinct coin types, where \( A_1 < A_2 < \ldots < A_n \). The coin-changing problem is defined as follows. Given an integer \( C \), find the smallest number of coins from \( A \) that adds up to \( C \), given that unlimited number of coins of each type is available. Design an efficient dynamic programming algorithm that on inputs \( A \) and \( C \), outputs the minimum number of coins needed to solve the coin-changing problem. State and prove the time complexity of your algorithm.

Problem 5 (20 points)

You are a melon-selling farmer with a collection of melons of distinct integral weights (you can assume they are sorted by weight). Customers come to you and ask for different integral weights of uncut melons. Devise an algorithm (as efficient as possible) that takes the customer’s request \( M \) as an input (\( M \) is an integer and is the weight of the uncut melons the customer is asking) and determines if it is possible to fulfill the customer’s request or not. Pay special attention while justifying its complexity (remember: your algorithm is fixed beforehand and should work for any \( M \), since \( M \) is part of the user input and not part of the algorithm).

Problem 6 (15 points)

You are a grad student in the Chemistry department, and you have to transport \( n \) chemicals in a box. Now, some chemicals react with some others (you know which ones react with which ones, and for simplicity, assume that every reaction requires exactly two chemicals to come in contact). Therefore, to separate the chemicals, you need to buy “separators” each of which effectively divides the space into two compartments. Now, it is obvious that if you have enough money to buy \( (n-1) \) separators, that would divide the box into \( n \) compartments, so you can easily make sure no reactions happen, even if every chemical reacts with every other. However, you are a poor grad student without funding, and you have enough money to buy only one separator. Design an algorithm that would determine if you can still transport all the chemicals without any reaction between them.

Problem 7 (10 points)

You are the general of a war. Your opponent has spread out his forces into \( N \) camps at various locations, and every camp is connected with every other camp through one or more supply chains (like roads, canals, air channels etc). These supply chains are very vital for the camps, since it is through these supply chains they communicate and move food & ammunition. You know the camp locations and the supply chain information.

You want to compile a list of all those supply chains so that if you destroy any of the supply chains in that list, the enemy force will immediately be separated into two disconnected components. Design an algorithm to achieve it.