Exercise 1 (10 points)

a) In the land of Oddaria, coins have weird integral values (not necessarily 1, 2, 5, etc.). 
Cashiers frequently have to solve the following problem when handing out change. Given a
positive integer \( t \) (the amount of change) and \( n \) positive integers \( w_1, \ldots, w_n \) (the values of all
coins at the cashier’s disposal), decide whether some subset of \( \{w_1, \ldots, w_n\} \) sums exactly to \( t \).
Design a fast dynamic programming algorithm and analyze its running time in terms of \( n \) and \( t \).

b) Coins are either rare or regular. Adapt your solution to give, among all ways of giving
change, the way that hands out the least number of rare coins.

Exercise 2 (8 points)

In the lecture we focused on polynomial multiplication when both polynomials have the same
degree \( d \). Generalize the results obtained in the lecture to multiplication of polynomials \( p(x) \) of
degree \( n \) and \( q(x) \) of degree \( m \). (You may use the algorithms obtained in the lecture as black
boxes.)

Exercise 3 (10 points)

a) Convolution: We are given vectors \( (a_0, a_1, \ldots, a_n) \) and \( (b_0, b_1, \ldots, b_m) \) with integer coeffi-
cients (each coefficient fits in a machine cell). We want to compute their convolution
\[(c_0, c_1, \ldots, c_{n+m}) \text{ where } c_i = \sum_j a_j \cdot b_{i-j}. \text{ Design an } O(n \log n) \text{ algorithm.}\]

b) **Boolean Convolution:** We are given bit-vectors \((a_0, a_1, \ldots, a_n)\) and \((b_0, b_1, \ldots, b_m)\). We want to compute their Boolean convolution \((c_0, c_1, \ldots, c_{n+m})\) where \(c_i = \bigvee_j (a_j \land b_{i-j})\) (that is, \(c_i\) is 1 if there exists a \(j\) such that \(a_j\) and \(b_{i-j}\) are both 1). Design an \(O(n \log n)\) algorithm.

**Exercise 4 (12 points)**

*Pattern matching with wildcards:* We are given a text string \(t\) of length \(n\) and a pattern string \(p\) of length \(m\). Both \(p\) and \(t\) contain symbols from an alphabet \(\Sigma\) of constant size. In addition, \(p\) may contain a wildcard character "*". We want to know whether any replacement of the wildcard characters in \(p\) by alphabet symbols yields a substring of \(t\). For example, \(p = \text{"bl**k"}\) matches two substrings of \(t = \text{"blank block"}\). Give an \(O(n \log n)\) time algorithm for this problem.

**Hint:** Use fast polynomial multiplication.