Total possible marks: 100. Homeworks must be solved individually. This set covers chapters 15–16 of the textbook *Introduction to Algorithms*, 3rd. ed., by Cormen et al.

Exercise 1: matrix-chain multiplication (25 points).

1. (15 points) Find an optimal parenthesization of a matrix-chain product whose sequence of dimensions is $\langle 5, 10, 3, 50, 6 \rangle$. Fill in the m and s tables as in fig. 15.3 in the textbook.
2. (10 points) Give the result obtained using a greedy algorithm that picks the smallest local cost $p_{i-1}p_kp_j$ over $i \le k < j$. Comment on whether this greedy algorithm always solves correctly the matrix-chain multiplication problem or not.

Exercise 2: coin changing problem (25 points). (This is based on problem 16-1 in the textbook.) Consider the problem of making change for n cents using the fewest number of coins.

1. (5 points) Describe in English a greedy algorithm to make change consisting of quarters, dimes, nickels and pennies.
2. (7 points) Write it in pseudocode, specifically a procedure Greedy-Change(n) that returns the number of quarters, dimes, nickels and pennies. Try it on $n = 89$ cents.
3. (7 points) Prove that it satisfies the greedy-choice property.
4. (6 points) Now we want to make change consisting only of quarters, dimes and pennies (nickels are not allowed). Apply the greedy algorithm to $n = 30$ cents and explain what happens. What kind of algorithm would you use to solve this coin changing problem? (you don't need to solve the problem itself).

Exercise 3: Huffman codes (25 points). The following table gives the frequencies of each character in a file with 1 000 characters.

α	β	γ	δ	ϵ	ζ	η	θ
400	100	300	50	80	10	20	40

1. (3 points) How many bits do we need to store the file if using a fixed-length code?
2. (8 points) Show the code built by the Huffman algorithm, both as a tree and as a list (character, codeword).
3. (3 points) How many bits do we need to store the file with this Huffman code?

Assume now that all the the characters have the same frequency (= 125).

4. (8 points) Show the code built by the Huffman algorithm, both as a tree and as a list (character,codeword).
5. (3 points) How many bits do we need to store the file with this Huffman code?

Exercise 4: 0–1 knapsack problem (25 points). Write the pseudocode for a dynamic programming algorithm that solves the 0–1 knapsack problem (either top-down with memoization, or bottom-up).

Bonus exercise 1: matrix-chain multiplication (15 points). (Exercise 15.2-2 in the textbook.) Give a recursive algorithm MATRIX-CHAIN-MULTIPLY(A, s, i, j) that actually performs the optimal matrix-chain multiplication, given the sequence of matrices $\langle A_1, A_2, \ldots, A_n \rangle$, the s table computed by procedure MATRIX-CHAIN-ORDER, and the indices i and j. (The initial call would be MATRIX-CHAIN-MULTIPLY$(A, s, 1, n)$.)

Bonus exercise 2: longest common subsequence (15 points). Construct the matrices c and b as in fig. 15.8 in the textbook for the words SPANKING and AMPUTATION.