Problem 8-1 (I Am a Big Fan of Rats)  

Using dynamic programming, find the optimum printing of the text “I am a big fan of rats”, i.e. \( \ell_1 = 1, \ell_2 = 2, \ell_3 = 1, \ell_4 = 3, \ell_5 = 3, \ell_6 = 2, \ell_7 = 4 \), with line length \( L = 10 \) and penalty function \( P(x) = x^3 \). Make sure you justify all your steps (and not just state the answer without proof). Will the optimal printing you get be consistent with the strategy “print the word on as long as it fits, and otherwise start a new line”?

Problem 8-2 (Plus/Minus/Zero)  

Consider \( n \) (decimal) digits \( a_1, \ldots, a_n \) (i.e., \( a_i \in \{0,1,\ldots,9\} \)) and an integer \( x \) with \( -9n \leq x \leq 9n \). An \( x \)-signing consists of \( n \) values \( s_1, \ldots, s_n \) with \( s_i \in \{1, 0, -1\} \) such that \( \sum_{i=1}^{n} s_i \cdot a_i = x \).

Your goal in this task is to devise a dynamic-programming algorithm that given \( a_1, \ldots, a_n \) and \( x \), decides whether there is an \( x \)-signing and, if so, outputs it. To that end,

- define an appropriate table,
- derive a recurrence,
- show how to efficiently fill the table, and
- devise a way to reconstruct the signing (if existent) from the filled table.

Analyze the running time of your algorithm.

Problem 8-3 (Who Needs a Nickel? Greedy People!)  

Assume you have \( q \) quarters, \( d \) dimes, \( n \) nickels and \( p \) pennies. You just bought an amazing textbook on basic algorithms and paid \( T > 0 \) cents for it. You wish to provide the exact change of \( T \) cents using the minimum number of coins \( c \). For example, if \( q = d = n = p = 2 \) and \( T = 27 \), the optimal solution is \( 25 + 1 + 1 = 27 \), using only \( c = 3 \) coins, since the other solution \( 10 + 10 + 5 + 1 + 1 \) uses 5 coins. On the other hand, if \( T = 23 \), there is no solution at all, since you only have 2 pennies.

(a) (10 points) Devise an \( O(T \cdot (q + d + n + p)) \) dynamic-programming algorithm that computes the minimum number of coins \( c \). If there is no solution, your algorithm should return \( c = \infty \).

For full credit, use only \( O(T) \) space. For partial credit, your space may be much worse, e.g., \( O(T \cdot q \cdot d \cdot n \cdot p) \). Note that you need not reconstruct the actual change!
(b) \( (4 + 1 \textbf{[Extra credit]} \text{ points}) \) Consider the following greedy algorithm: “Find the largest coin (among those you still have) whose value \( v \) is less than or equal to \( T \). Add this coin to your solution and recurse on \((T - v)\).” Give an example (i.e., a choice of \( T, q, d, n, p \)) where the greedy algorithm above fails to find the optimum solution. For extra credit, the greedy algorithm in your example should return \( c < \infty \). \textbf{(Hint: See the title of the problem!)}

**Problem 8-4 (Let’s paint a fence!)** 12 (+3) Points

We just built a new fence which is made up of many boards, and our task is to paint the fence efficiently. To paint the fence, you have to paint \( N \) boards that make up the fence. The lengths of the \( N \) boards are \( \{L_1, L_2, \ldots, L_N\} \). You have hired \( K \) painters and you know that each painter takes 1 hour to paint 1 unit of board. If \( 3, 4, 5 \) are the boards painter \( i \) paints, the total time he spends is \( t_i = L_3 + L_4 + L_5 \). Our goal is to assign each painter to boards so that the total painting time is minimized. Since the painters can work in parallel, the painting time is minimized when \( \max(t_1, \ldots, t_K) \) is minimal. The painting task must be accomplished under the following constraints:

1. Two painters cannot share a board to paint. That is, a board cannot be painted partially by one painter, and partially by another. All \( L_i \) units of board \( i \) must be painted by one painter.

2. Any painter will only paint contiguous boards. For example, a configuration where painter 1 paints boards 1 and 3 but not 2 is not a valid solution.

You are given as input the following: \( K \), the number of painters, and \( L \), a list which will represent the length of each board, where \( L_i \) is the length of the \( i \)th board. In the following problems, denote by \( T[i, j] \) the minimum time to paint the first \( i \) boards with \( j \) painters. We will, successively in the following subtasks, come up with a procedure to minimize painting time.

(a) \( (6 \text{ points}) \) Write a recurrence for \( T[i, j] \) in terms of \( T[* , j - 1] \).

(b) Note that the time required to fill table \( T \) is \( O(NK \cdot C) \), where \( C \) is the maximum time needed to compute any of the entries \( T[i, j] \). Provide the fastest algorithm you can conceive. You will obtain

- 2 points if your algorithm satisfies \( C = O(N^2) \),
- an additional 4 points if \( C = O(N) \),
- and 3 points of \textbf{extra credit} if you manage to get it down to \( C = O(\log N) \).