Problem 1 (20 points). Let $S_1 = AGCCTG$ and $S_2 = AGCT$ be two strings over alphabet $\Sigma = \{A, C, G, T\}$. Draw the dynamic programming table for computing the edit distance between S_1 and S_2.

Problem 2 (20 points). A subsequence is *palindromic* if it is the same whether read left to right or right to left. Given a string A of length n, design a dynamic programming algorithm to find the longest palindromic subsequence of A. Your algorithm should run in $O(n^2)$ time. For example, the longest palindromic subsequence for string $A = ACTCGCATA$ is $ATCGCTA$.

Problem 3 (20 points). Let $A[1 \cdots n]$ be an array with n integers (possibly with negative ones). Design a dynamic programming algorithm to compute indices i and j such that $\sum_{k:i \le k \le j} A[k]$ is maximized. Your algorithm should run in $O(n)$ time.

Problem 4 (20 points). Let $A = a_1 a_2 \cdots a_n$ and $B = b_1 b_2 \cdots b_m$ be two strings. Design a dynamic programming algorithm to compute the *longest common substrings* between A and B, i.e., to find the *largest* k and indices i and j such that $A[i \cdots i+k] = B[j \cdots j+k]$. Your algorithm should run in $O(mn)$ time.

Problem 5 (20 points). Let $A = a_1 a_2 \cdots a_n$ and $B = b_1 b_2 \cdots b_m$ be two strings. Design a dynamic programming algorithm to compute the *longest common subsequence* between A and B, i.e., to find the *largest* k and indices $1 \le i_1 < i_2 < \cdots < i_k \le n$ and $1 \le j_1 < j_2 < \cdots < j_k \le m$ such that $A[i_1] = B[j_1], A[i_2] = B[j_2], \cdots, A[i_k] = B[j_k]$. Your algorithm should run in $O(mn)$ time.