Problem 1 (10 points).

Solve the "Hidden Surface Removal" problem (Problem 5, Chapter 5 on page 248 of the Textbook [KT]) using certain algorithm(s) introduced in our lectures.

Problem 2 (15 points).

You are given n points $P = \{p_1, p_2, \cdots, p_n\}$ on 2D plane, represented as their coordinates. You are informed that the convex hull of P contains exactly 8 points in P (assume that $n \geq 8$). Design an algorithm to compute the convex hull of P in $O(n)$ time. You may assume that no three points in P are on the same line.

Problem 3 (15 points).

You are given a non-convex polygon represented with its vertices $P = \{p_1, p_2, \cdots, p_n\}$ in couter-clockwise order. Design an algorithm to find the convex hull of P in $O(n)$ time. Prove that your algorithm is correct.

Problem 4 (20 points).

You are given two sets of points $P = \{p_1, p_2, \cdots, p_n\}$ and $Q = \{q_1, q_2, \cdots, q_n\}$ on 2D plane. Design an algorithm to determine whether there exists a non-vertial line L such that all points in P locate in one side of L while all points in Q locate in another side of L. Your algorithm should run in $O(n \cdot \log n)$ time.

Problem 5 (20 points).

Given an array A with n positive distinct integers, design an algorithm to count the number of pairs $(A[i], A[j])$ satisfying that $2 \cdot i \leq j$ and $A[i] \geq 2 \cdot A[j]$. Your algorithm should run in $O(n \cdot \log n)$ time.

Problem 6 (15 points).

Let $G = (V, E)$ be a directed graph with positive edge length $l(e) > 0$ for any $e \in E$. For vertex $v \in V$ there is also an associated positive *vertex weight* $w(v) > 0$. For any path we define its *length* as the sum of the length of its all edges plus the sum of the weights of its all vertices. Given $s \in V$, design an algorithm runs in $O(|V| \cdot |E|)$ time to find the shortest paths (with the new definition of length) from s to all other vertices.

Problem 7 (30 points).

Let $c_1, c_2, \cdots, c_n$ be various currencies. For any two currencies c_i and c_j, there is an exchange rate $r_{i,j}$; this means that you can purchase $r_{i,j}$ units of currency c_j in exchange for one unit of c_i. These exchange rates satisfy the condition that $r_{i,j} \cdot r_{j,i} < 1$, so that if you start with a unit of currency c_i, change it into currency c_j and then convert back to currency c_i, you end up with less than one unit of currency c_i (the difference is the cost of the transaction). Give an efficient algorithm for the following problem: given a set of exchange rates $r_{i,j}$, and two currencies s and t, find the most advantageous sequence of currency exchanges for converting currency s into currency t.

Occasionally the exchange rates satisfy the following property: there is a sequence of currencies $c_1, c_2, \cdots, c_k$ such that $r_{1,2} \cdot r_{2,3} \cdot r_{3,4} \cdots r_{k-1,k} \cdot r_{k,1} > 1$. This means that by starting with a unit of currency c_1 and then successively converting it to currencies c_2, c_3, $\cdots$, c_k, and finally back to c_1, you would end up with more than one unit of currency c_1. Give an efficient algorithm for detecting the presence of such an anomaly.

Problem 8 (20 points).

In each of a continuous period of n days, an easy task and a hard task is given. You can choose to work on the easy task, which gives you reward a_k, $1 \leq k \leq n$, work on the hard task, which gives you reward b_k, $1 \leq k \leq n$, or choose neither of them, which of course gives you reward 0. If you choose to work on a hard task on day k, $1 \leq k < n$, then you need to rest on the next day, i.e., choose to work on neither of the tasks

on day $(k+1)$. Given a_k and b_k, $1 \leq k \leq n$, design a dynamic programming algorithm to schedule the task on each day so that the total rewards is maximized. Define the subproblems, give the recursion, describe the initialization, iteration, and termination steps of the algorithm, and give the running time of your algorithm.

Problem 9 (20 points).

A string 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. (For example, the longest palindromic subsequence for string $A = ACTCGCATA$ is $ATCGCTA$.) Define the subproblems, give the recursion, describe the initialization, iteration, and termination steps of the algorithm, and give the running time of your algorithm. Your algorithm should run in $O(n^2)$ time.