ALGORITHMS (50%)

1. A complete binary tree is defined inductively as follows. A complete binary tree of height 0 consists of 1 node which is the root. A complete binary tree of height h+1 consists of two complete binary trees of height h whose roots are connected to a new root. Let T be a complete binary tree of height h. The height of a node in T is h minus the node's distance from the root (e.g., the root has height h, whereas a leaf has height 0). What is the sum of the heights of all the nodes in T? (10%)
2. Given a sequence of real numbers a_n, a_{n-1}, …, a_1, a_0, and a real number x, please design a most efficient algorithm (the Horner's rule) to compute the value of the polynomial $P_n(x) = a_n x^n + a_{n-1}x^{n-1} + \ \ldots + a_1 x + a_0$. You must analyze and show how many multiplications and additions are required in the algorithm. (10%)
3. (a) What is the knapsack problem? (5%)
   (b) What is dynamic programming? Give an example. (5%)
4. The towers of Hanoi puzzle is a well-known problem. There are n disks of different sizes arranged on a peg (named A) in decreasing order of sizes. There are two other empty pegs (named B and C). The purpose of the puzzle is to move all the disks, one at a time, from the first peg (A) to another peg (C) in the following way. Disks are moved from the top of one peg to the top of another. A disk can be moved to a peg only if it is smaller than all other disks on that peg. In other words, the ordering of disks by decreasing sizes must be preserved at all times. The goal is to move all the disks in as few moves as possible. (10%)
   (a) Design a recursive algorithm to find a **minimal** sequence of moves that solves the towers of Hanoi problem for n disks.
   (b) How many moves are used in your algorithm?
5. Define **P** problem, **NP** problem, **NP-hard** problem, and **NP-complete** problem. Give an example that is a **NP-complete** problem. (10%)

(背面仍有題目,請繼續作答)

Data Structure (50%)

6. (a) Define a data structure to store a given 2-D polygon (封閉多邊形). (5%)

   (b) Write a pseudo code to test if a given 2-D polygon is convex(凸邊形) or concave (凹邊形). (5%)

   (c) Write a pseudo code to test if a given 2-D point is inside or outside a 2-D polygon. The 2-D polygon can be convex or concave. (10%)
7. For two given N x N matrixes, write a recursive pseudo code to compute their multiplication (15%)
8. Write a pseudo code to simulate a recursive function call by using stack structure. Please use a simple recursive program to explain your answer. (15%).

Note: your score depends on the correctness and logic. Please answer above problems clearly and formally.