If some questions are unclear or not well-defined to you, you can make your own assumptions and state the reasons clearly in the answer sheet.

1. Let $P = (p_1, p_2, ..., p_n)$ be a permutation of $\{1,2, ..., n\}$. b_i, $2 \le i \le n$, denotes a permutation operator that $b_i(P) = (p_1, p_2, ..., p_{i-3}, p_{i-2}, p_i, p_{i-1}, p_{i+1}, p_{i+2}, ..., p_n)$. For example, $b_3(13254) = (12354)$ and $b_5(13254) = (13245)$. A subset of the permutation operators b_i's can be used to sort a permutation to the identity permutation, which is $(1,2,3,...,n)$. The *bubble sort* uses a subset of b_i's to perform sorting. For example, we can sort (13254) by applying b_3 followed by b_5, or b_5 followed by b_3. Now, a new set of permutation operators k_i's, $2 \le i \le n$, are defined as that $k_i(P) = (p_i, p_{i-1}, ..., p_3, p_2, p_1, p_{i+1}, p_{i+2}, ..., p_n)$. For example, k_3(13254)= (23154) and k_5(13254)=(45231). Answer the following questions by applying k_i, $2 \le i \le n$.
   (a) How do you sort (23514) to (12345) with minimum number of steps? Please give the sequence of k_i's you use in this sorting. (5%)
   (b) Give a general algorithm which sorts any permutation $P = (p_1, p_2, ..., p_n)$ to the identity permutation $(1,2,3,...,n)$. (5%)
   (c) In the worst case, how many steps are needed for the above algorithm? Give your reasons. (5%)
2. A *subsequence* of a given string A can be obtained by deleting 0 or more symbols from A (not necessarily consecutive). For example, "acad" and "bcd" are two of subsequences of "bacad". The *longest common subsequence* (LCS) of two strings A and B is a subsequence that appears in both A and B, and it is the longest. The LCS problem is to find the length of the LCS of two given strings. Let $A = a_1 a_2 ... a_m$ and $B = b_1 b_2 ... b_n$. Let $L_{i,j}$ denote the length of the LCS of $a_1 a_2 ... a_i$ and $b_1 b_2 ... b_j$. We can use the following dynamic programming to calculate $L_{i,j}$:

   $L_{0,0} = L_{0,j} = L_{i,0} = 0,\quad$ for $1 \le i \le m, 1 \le j \le n$,

   $L_{i,j} = L_{i-1,j-1} + 1\quad$ if $a_i = b_j$, for $1 \le i \le m, 1 \le j \le n$,

   $L_{i,j} = max\{ L_{i-1,j}, + L_{i,j-1}\}\quad$ if $a_i \ne b_j$, for $1 \le i \le m, 1 \le j \le n$. Now, we

   have another problem, *the edit distance problem*. The costs of each replacement, insertion and deletion are assumed to be 1, 2 and 2, respectively. The total edit cost of two strings is the sum of those operation costs performed on one string to become another. For example, A= "vint" and B= "write", we need 1 replacement, 2 insertions and 1 deletion to change A to B. The total edit cost is 7.
   (a) Please point out the relationship between the longest common subsequence problem and the edit distance problem. (5%)

(b) Design an algorithm to find the minimum total edit cost between two strings. (10%)

(c) Give the time complexity of your above algorithm. (5%)

3. A *strictly binary tree* is a binary tree in which every nonleaf node has nonempty left and right subtrees. Write a C or Pascal program to decide whether a binary tree is strictly binary or not, and to calculate the sum of the values stored in the tree. Please point out which language (C or Pascal) you are using to write the program. You have to give your declaration in your program completely. (15%)

4. (a) You are going to enhance a machine and there are two possible improvements: either make multiply instructions run four times faster than before, or make memory access instructions run two times faster than before. You repeatedly run a program that takes 100 seconds to execute. Of this time, 20% is used for multiplication, 50% for memory access instructions, and 30% for other tasks. What will the speedup be if you improve only multiplication? What will the speedup be if you improve only memory access? What will be the speedup be if both improvements are made? (5%)

(b) You are going to change the program described in problem 1.1 so that the percentages are not 20%, 50%, and 30% anymore. Assuming that none of the new percentages is 0, what sort of program would result in a tie (with regard to speedup) between the two individual improvements? Provide both a formula and some examples. (5%)

5. A full adder can be thought as a hardware device that can add three inputs together (*ai, bi, ci*) and produce two outputs (*s, ci+1*). When we are adding more than two numbers, it is possible to reduce the cost of the carry using the concept of *carry-save addition*. This idea is to form two independent sums, called S' (sum bits) and C' (carry bits). At the end of the process, we need to add C' and S' together using a normal adder. For example, adding four numbers using carry save addition needs two levels of carry save adders followed by a normal adder.

(a) Calculate the delays and area costs to add four 16-bit numbers using pure ripple carry adders, versus carry save addition with a ripple carry adder forming the final sum. (Assume that the delay through a 1-bit full adder cell is T_{FA} and the area of a 1-bit full adder cell is A_{FA}.) (5%)

(b) Generalize the above result by compare the delays and area costs to add 10 *n*-bit numbers. (5%)

6. RAM chips are sometimes designed with two chip select lines, CS1 and CS2, both of which must be enabled, that is, CS1=CS2=logic 1, to initiate a read or write operation. This makes it possible to arrange a memory as a two-dimensional array of ICs, in which each IC has a row address enabled by CS1 and a column address

enabled by CS2. Explain this technique for the design of a 64M x 8 RAM that is to be constructed from 1M x 8 RAM ICs with two CS lines and draw the array design of the memory using decoders/multiplexers and the composing RAM ICs. (15%)

7. Explain the following terminologies: (15%)
   (a) Context Switching
   (b) Control Hazard, Data Hazard
   (c) Forwarding
   (d) Dispatch
   (e) Reorder Buffer