Midterm

Note

This is a closed-book exam. Each problem accounts for 10 points, unless otherwise marked.

Problems

1. Let $a_1, a_2, \cdots, a_n$ be positive real numbers such that $a_1 a_2 \cdots a_n = 1$. Prove *by induction* that $(1 + a_1)(1 + a_2) \cdots (1 + a_n) \geq 2^n$. (Hint: In the inductive step, try introducing a new variable that replaces two chosen numbers from the sequence.)

2. Consider a round-robin tournament among n players. In the tournament, each player plays once against all other $n - 1$ players. There are no draws, i.e., for a match between A and B, the result is either A beat B or B beat A. Prove *by induction* that, after a round-robin tournament, it is always possible to arrange the n players in an order $p_1, p_2, \cdots, p_n$ such that p_1 beat p_2, p_2 beat p_3, $\cdots$, and p_{n-1} beat p_n. (Note: the "beat" relation, unlike "$\geq$", is not transitive.)

3. Below is an algorithm for solving a variant of the Towers of Hanoi puzzle with an additional fourth peg D; `Towers_Hanoi` is an algorithm for the original puzzle.

```
Algorithm Four_Towers_Hanoi(A,B,C,D,n);
begin
   if n<=2 then
      Towers_Hanoi(A,B,C,n);
   else
      Four_Towers_Hanoi(A,D,B,C,n-2);
      Towers_Hanoi(A,B,C,2);
      Four_Towers_Hanoi(D,B,C,A,n-2);
end;
```

   Consider alternatives of first moving $n-k$ disks (for some value of k, not necessarily 2) to D. Let $T(n)$ denote the number of moves needed for n disks. Write a recurrence relation for $T(n)$ with k as a parameter. Can you tell which value of

k will be the best, i.e., resulting in a smaller asymptotic upper bound for $T(n)$? Why?

4. In the implementation of an AVL tree, a rebalancing process using rotation operations may be needed after an `insert` or `delete`. Design the first part of a procedure for `insert`, up to the point when the node where a rotation is needed (i,e., the *critical* node) is determined (when rebalancing is needed). You are not required to design the part for rotation. Please present your procedure in an adequate pseudo code and make assumptions wherever necessary.

5. Show all intermediate and the final AVL trees formed by inserting the numbers 4, 5, 6, 1, 2, and 3 (in this order) into an empty tree. Please use the following ordering convention: the key of an internal node is larger than that of its left child and smaller than that of its right child. If a rotation is performed during an insertion, please also show the tree before the rotation.

6. Design an efficient algorithm that, given an array A of n integers and an integer x, determine whether A contains two integers whose sum is exactly x. Please present your algorithm in an adequate pseudo code and make assumptions wherever necessary. Give an analysis of its time complexity. The more efficient your algorithm is, the more points you will be credited for this problem.

7. The Knapsack Problem is defined as follows: Given a set S of n items, where the ith item has an integer size $S[i]$, and an integer K, find a subset of the items whose sizes sum to exactly K or determine that no such subset exists.

   Below is an algorithm for determining whether a solution to the problem exists.

```
Algorithm Knapsack (S, K);
begin
    P[0,0].exist := true;
    for k := 1 to K do
        P[0,k].exist := false;
    for i := 1 to n do
        for k := 0 to K do
            P[i,k].exist := false;
            if P[i-1,k].exist then
               P[i,k].exist := true;
               P[i,k].belong := false
            else if k - S[i] ≥ 0 then
```

if $P[i-1, k-S[i]].exist$ **then**
$P[i,k].exist := true;$
$P[i,k].belong := true$

end

(a) Design an algorithm to recover the solution recorded in the array P. (5 points)

(b) Modify the given algorithm to solve a variation of the knapsack problem where each item has an unlimited supply. (10 points)

8. Let $x_1, x_2, \cdots, x_n$ be a sequence of real numbers (not necessarily positive). Design an $O(n)$ algorithm to find the subsequence $x_i, x_{i+1}, \cdots, x_j$ (of consecutive elements) such that the product of the numbers in it is maximum over all consecutive subsequences. The product of the empty subsequence is defined to be 1. Please present your algorithm in an adequate pseudo code and make assumptions wherever necessary.

9. Consider rearranging the following array into a max heap.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
8	2	3	5	12	14	7	1	6	4	10	15	13	9	11

(a) Design a systematic procedure for performing this task. Please present your procedure in an adequate pseudo code and make assumptions wherever necessary. (10 points)

(b) The procedure above most likely will consist of a number of rounds. Given the above input, please show the result (i.e., the contents of the array) after each round until it becomes a heap. (5 points)