Midterm

Note

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

Problems

1. 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;
```

   Let $T(n)$ denote the number of moves needed for n disks. Write a recurrence relation for $T(n)$ and solve it.

2. Consider binary trees where each node stores a non-negative integer. Design an algorithm that, given such a tree T and a non-negative integer k as input, determines whether T contains a branch (from the root to a leaf) such that the sum of all numbers stored on the nodes of the branch equals k. The more efficient your algorithm is, the more points you will be credited for this problem. Is there a possibility that your code may overflow? Have you avoided the problem? (15 points)

3. Modify the following code for determining the sum of the maximum consecutive subsequence so that it also records the start and end indices of the subsequence.

```
Algorithm Max_Consec_Subseq (X, n);
begin
    Global_Max := 0;
    Suffix_Max := 0;
    for i := 1 to n do
        if x[i] + Suffix_Max > Global_Max then
            Suffix_Max := Suffix_Max + x[i];
            Global_Max := Suffix_Max
        else if x[i] + Suffix_Max > 0 then
                Suffix_Max := Suffix_Max + x[i]
        else Suffix_Max := 0
end
```

4. Show all intermediate and the final AVL trees formed by inserting the numbers 4, 2, 1, 0, 7, 8, 9, 5, 6, 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. (15 points)

5. Please present the union-find algorithm with balancing and path compression in a suitable pseudocode. (20 points)

6. Rearrange the following array into a (max) heap using the bottom-up approach.

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

   Show the result after each element is added to the part of array that already satisfies the heap property.

7. Design an algorithm that determines whether two sets of numbers (represented as arrays) are disjoint; the more efficient your algorithm is, the more points you will be credited for this problem. State the time complexity of your algorithm in terms of the sizes m and n of the given sets. Be sure to consider the case where m is substantially larger than n.

8. Draw a Huffman tree for a text with the following frequency distribution: $A : 12$, $B : 7$, $C : 6$, $D : 4$, $E : 15$, $F : 4$, $G : 3$, and $H : 2$.