QUESTION 1. [10 pts] Consider the following algorithm

Algorithm Mystery(A[1..n]: integer);
  var i, j: integer;
  y = 1;
  for i := 1 to n do
    for j := 1 to 2*i do
      y := y + y;
    i := n;
  while i > 1 do
    y := y + 1;
    i := i/2;
    for j := 1 to n do
      y := y * y;
  print y;

Let $T(n)$ denote the (worst-case) time complexity of algorithm Mystery. Analyze the algorithm to obtain a tight asymptotic bound on $T(n)$. You need show the key steps in your analysis.

$$T(n) = \sum_{i=1}^{n} \Theta(i) + \Theta(n \log n)$$

$$= \Theta(n^2) + \Theta(n \log n)$$

$$= \Theta(n^2)$$
QUESTION 2. [10 pts] Consider the following recursive algorithm

Algorithm Sillyaverage(var B[1..n]: integer);
    var i,j: integer;
    begin
        if n > 1 then
            for i := 1 to n-1 do
                B[i+1] := (B[i] + B[i+1]) / 2;
            call Sillyaverage(B[1..n-1])
        end;

Let $T(n)$ denote the (worst-case) time complexity of algorithm Sillyaverage. Use recurrence relations to obtain a tight asymptotic bound on $T(n)$.

$$T(n) = T(n-1) + \Theta(n)$$

$$= T(n-1) + \Theta(n-1) + \Theta(n)$$

$$\cdots$$

$$= T(1) + \sum_{i=2}^{n} \Theta(i)$$

$$= 1 + \sum_{i=2}^{n} \Theta(i)$$

$$= \sum_{i=1}^{n} \Theta(i)$$

$$= \Theta(n^2)$$

```
Algorithm PrefixSum(A[1..n]: integer);
    var i, j: integer;
    begin
        for i := 1 to n do
            B[i] = A[i];
        for j := 2 to i do
    end;
```

(a) [5 pts] What is the time complexity of algorithm PrefixSum?

(b) [10 pts] Can you design another algorithm with an improved time complexity? Give the (informal) pseudocode and analyze its time complexity.

(a) $T(n) = \sum_{i=1}^{n} i = \Theta(n^2)$

(b) Alg. ImprovedPrefixSum (A[1..n]: integer);
    var i, j: integer;
    begin
        B[0] := A[i];
        for i := 2 to n do
    end.

Time: $\Theta(n)$
QUESTION 4. [10 pts] Let $A[1..8]$ be an array of eight integers. Describe how to find both the maximum and minimum elements in $A$ using at most 10 comparisons.

*Hint:* Divide-and-conquer. Can you solve the problem for an array of four integers in 4 comparisons?

You may describe your solution by means of an informal pseudocode or a (branching) diagram. Indicate the number of comparisons at each step.

Algorithm $\text{MinMax}(A[1..n])$

if $n > 1$ then

$(\text{min}_1, \text{max}_1) := \text{MinMax}(A[\frac{n}{2}+1..n])$

$(\text{min}_2, \text{max}_2) := \text{MinMax}(A[1..\frac{n}{2}])$

\[ \text{min} := \min\{\text{min}_1, \text{min}_2\} \]

\[ \text{max} := \max\{\text{max}_1, \text{max}_2\} \]

return $(\text{min}, \text{max})$

OR

\[ a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8 \]

\[ \text{min, max, min}_2, \text{max}_2 \]

\[ \text{min} \]

\[ \text{max} \]

\[ \text{min} \]

\[ \text{max} \]

4 comp. 4 comp. 2 comp.

10 pts
QUESTION 5. [15 pts] Recall that in the Simplified Fake Coin problem we are given a set of $n$ coins and a balance scale. Suppose we know that there is a fake coin among the $n$ coins that is lighter than the genuine one. Design an algorithm (called the divide-into-three algorithm in Levitin) to find the fake coin in at most $\log_3 n$ steps.

(a) [8 pts] Give an (informal) pseudocode of your algorithm. For simplicity, you may assume that $n = 3^k$ for some integer $k$.

(b) [7 pts] Analyze the time complexity of your algorithm by first setting up a recurrence relation and then solving it for $n = 3^k$.

(a) Alg. Fast Coin ($S$);
   
   \hspace*{1cm} // $S$ is a set of $n$ coins //

   \hspace*{1cm} Divide $S$ into $S_1$, $S_2$, $S_3$

   \hspace*{1cm} with equal sizes,

   \hspace*{1cm} if $W(S_1) = W(S_2)$ then

   \hspace*{1cm} Fast Coin ($S_3$)

   \hspace*{1cm} else if $W(S_1) < W(S_2)$ then

   \hspace*{1cm} Fast Coin ($S_1$)

   \hspace*{1cm} else

   \hspace*{1cm} Fast Coin ($S_2$)

(b) $T(n) = T(n/3) + 1$

   $= \log_3 n$ steps