Problem 1: Asymptotics [20 points].

(a) Prove or disprove: If \( f(n) \) is \( O(g(n)) \) then \( g(n) \) is \( \Omega(f(n)) \). [5 points]

(b) Prove or disprove: \( f(n) \) is \( O(g(n)) \) or \( g(n) \) is \( O(f(n)) \). [5 points]

(c) Prove or disprove: For every \( k \geq 2 \), \( \log_k(n) \) is \( O(\log_2(n)) \). [5 points]

(d) Prove or disprove: If \( f(n) = O(g(n)) \), then \( g(n) - f(n) = \Theta(g(n)) \). [5 points]

Problem 2: Cryptography (Taken from DPV 1.45) [15 points]. Recall that in the RSA public-key cryptosystem, each user has a public key \( P = (N, e) \) and a secret key \( d \). In a digital signature scheme, there are two algorithms: sign and verify. The sign procedure takes a message and a secret key, then outputs a signature \( \sigma \). The verify procedure takes a public key \( (N, e) \), a signature \( \sigma \), and a message \( M \), then returns “true” if \( \sigma \) could have been created by sign (when called with message \( M \) and the secret key corresponding to the public key \( (N, e) \)); “false” otherwise.

(a) Assume that Alice is trying to send a message to Bob, but the malicious Mallory may be trying to interfere with their communications. Explain in a sentence or two why Alice and Bob might want to use a digital signature scheme. [5 points]

(b) In an RSA digital signature scheme, we simply implement \( \text{sign}(M, d) = \sigma = M^d \mod N \), where \( d \) is the secret key of the party doing the signing and \( N \) is part of the public key of the party doing the signing. Give a corresponding verify procedure with the behavior outlined above. Argue that your verify procedure is correct. [10 points]
Problem 3: Recurrence Relations [10 points].

(a) Solve the following recurrence relation. Assume \( T(1) = 1 \). Express your answer in big-\( \Theta \) notation.
\[
T(n) = 2T(n - 1) + 2^n. \quad [5 \text{ points}]
\]

(b) \( T(n) = 2T(\sqrt{n}) + 1 \) with base \( T(2) = 1 \). (To simplify the solution, suppose you can always take the square root without a remainder in each step of the recurrence). [5 points]

Problem 4: Divide and Conquer [20 points]. Consider the following problem of covering a chess board board: You are given a \( 2^n \times 2^n \) size chess board with 1 arbitrary tile already covered and an unlimited number of “trominoes” which cover 3 tiles of the board in an L shape. An example of an \( n = 3 \) board is given in the figure below where the red tile is the already covered tile.

![Chess Board with Tromino](image)

Write a recursive algorithm in pseudocode to cover the rest of the board. Argue for the correctness of your algorithm. Infer a recurrence relation for your algorithm, solve that recurrence relation, and give the asymptotic running time of your algorithm.

Problem 5: More Divide and Conquer [20 points]. Given two sorted arrays of integers \( A \) and \( B \), each of size \( n > k \), write an algorithm that finds the \( k \)th smallest element of the union of the two arrays (for example, if \( A = [0, 1, 5] \) and \( B = [2, 3, 6] \) then the 2nd smallest element of their union would be 1). Argue for the correctness of your algorithm. Infer a recurrence relation for your algorithm, solve that recurrence relation, and give the asymptotic running time of your algorithm. For full credit, your algorithm should run in \( O(\log k) \) time.