Please read these notes

- This exam is closed book, closed notes and 50 minutes long
- Read the questions carefully
- No electronic equipment allowed (smart phones, tablets, computers, …)
- Write legibly and try to be brief and to the point; what can’t be read will not be graded
- No code: use pseudocode or English to describe your algorithms
- When designing an algorithm, you are allowed to use any algorithm or data structure we explained in class or covered in a CS 218 homework or exam, without giving its details, unless the question specifically requires that you give such details
- Always remember to analyze the time complexity of your solution
- If you have a question, please raise your hand
Problem 1. [Analysis][Lower bounds] (25 points)

Consider the following multi-search problem. Let \( A[1, \ldots, n] \) be an array of distinct integers. Given an array \( X[1, \ldots, k] \), we want to find the position of each integer \( X[i] \) in the array \( A \). In other words, we want to compute an array \( I[1, \ldots, k] \) where for each \( i \), either \( I[i] = 0 \) if \( X[i] \) does not appear in \( A \) or \( I[i] > 0 \) if \( A[I[i]] = X[i] \). For instance, if \( A = [2, 4, 5, 1] \) and \( X = [4, 6, 1] \), the algorithm should produce \( I = [2, 0, 4] \), because 4 is at position 2 in \( A \), 6 does not exist in \( A \), and 1 is at position 4 in \( A \). Provide a lower bound on the multi-search problem, as a function of \( n \) and \( k \), assuming the decision-tree (comparison-based) model of computation.
Problem 2. [Analysis][Amortized Analysis] (25 points)

An ordered stack is a data structure that stores integers and supports the following operations.

- **ORDEREDPush**(x) first removes integers from the top of the stack as long as they are smaller than x (i.e., it stops popping elements as soon as the integer on top of the stack is greater than or equal to x), then pushes x onto the stack

- **POP()** removes and returns the item on the top of the stack (or returns **Null** if the stack is empty).

Suppose we implement an ordered stack with a simple linked list, using the obvious **ORDEREDPush** and **POP** algorithms. Use the accounting method to design a charging scheme to prove that an arbitrary sequence of n these two operations starting from an empty stack will take \(O(n)\) overall. Make sure you explain the credit invariant.
Problem 3. [Divide and Conquer][Design] (25 points)

The Hadamard matrices $H_0, H_1, H_2, \ldots$ are defined as follows.

$$H_k = \begin{cases} [1] & k = 0 \\ \begin{bmatrix} H_{k-1} & H_{k-1} \\ H_{k-1} & -H_{k-1} \end{bmatrix} & k > 0 \end{cases}$$

Note that $H_k$ is a $2^k \times 2^k$ matrix. Design a $O(n \log n)$ divide-and-conquer algorithm that given a column vector $v$ of length $n = 2^k$, computes the matrix-vector product $H_k v$. Analyze the time complexity of your algorithm.
Problem 4. [Divide and Conquer][Knowledge] (25 points)

Let \( a = [a_0, a_1, \ldots, a_{n-1}] \) and \( b = [b_0, b_1, \ldots, b_{n-1}] \) two \( n \)-bits binary vectors. Explain how to compute the product \( a \times b \) in \( O(n \log n) \)-time as we discussed in class. Make sure you explain the time complexity of each step of the algorithm.