Name:

Student ID #:

- You are expected to work on this assignment on your own
- Use pseudocode, Python-like or English to describe your algorithms. Absolutely no C++/C/Java
- When designing an algorithm, you are allowed to use any algorithm or data structure we explained in class, without giving its details, unless the question specifically requires that you give such details
- Always remember to analyze the time complexity of your algorithms
- Homework has to be submitted electronically on Gradescope by the deadline. No late assignments will be accepted
Problem 1. (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.

Answer:
Problem 2. (25 points)

You are given two sorted list of size $m$ and $n$. Give a $O(\log k)$ time algorithm for computing the $k$-th smallest element in the union of the two lists. Note: Observe that the $k$-th smallest element in the union of the arrays $a[1\ldots m]$ and $b[1\ldots n]$ has to be contained in $a[1\ldots k]$ or $b[1\ldots k]$.

Answer:
Problem 3. (25 points)

Describe and analyze an algorithm that takes an unsorted array $A$ of $n$ integers (in an unbounded range) and an integer $k$, and divides $A$ into $k$ equal-sized groups, such that the integers in the first group are lower than the integers in the second group, and the integers in the second group are lower than the integers in the third group, and so on. For instance if $A = \{4, 12, 3, 8, 7, 9, 10, 20, 5\}$ and $k = 3$, one possible solution would be $A_1 = \{4, 3, 5\}, A_2 = \{8, 7, 9\}, A_3 = \{12, 10, 20\}$. Sorting $A$ in $O(n \log n)$-time would solve the problem, but we want a faster solution. The running time of your solution should be bounded by $O(nk)$. For simplicity, you can assume that is $n$ a multiple of $k$, and that all the elements are distinct. **Note:** $k$ is an input to the algorithm, not a fixed constant.

Answer:
Problem 4. (25 points)

Given an array of numbers $X = \{x_1, x_2, \ldots, x_n\}$, an *exchanged pair* in $X$ is a pair $(x_i, x_j)$ such that $i < j$ and $x_i > x_j$. Note that an element $x_i$ can be part of up to $n - 1$ exchanged pairs, and that the maximal possible number of exchanged pairs in $X$ is $n(n-1)/2$, which is achieved if the array is sorted in descending order. Develop a divide-and-conquer algorithm that counts the number of exchanged pairs in $X$ in $O(n \log n)$ time. Argue why your algorithm is correct, and why your algorithm takes $O(n \log n)$ time. You can assume that $n$ is a power of two.

Answer: