Question 1: The Hadamard matrices $H_0, H_1, H_2, \ldots$ are recursively defined as follows:

$$H_0 = (1)$$

and for $k \geq 1$,

$$H_k = \begin{pmatrix} H_{k-1} & H_{k-1} \\ H_{k-1} & -H_{k-1} \end{pmatrix}.$$

Thus, $H_0$ is a $1 \times 1$ matrix whose only entry is 1,

$$H_1 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix},$$

and

$$H_2 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}.$$

Observe that $H_k$ has $2^k$ rows and $2^k$ columns.

If $x$ is a column vector of length $2^k$, then $H_kx$ is the column vector of length $2^k$ obtained by multiplying the matrix $H_k$ with the vector $x$.

Describe a recursive algorithm $\text{MULT}(k, x)$ that does the following:

**Input:** An integer $k \geq 0$ and a column vector $x$ of length $n = 2^k$.

**Output:** The column vector $H_kx$ (having length $n$).

The running time $T(n)$ of your algorithm must be $O(n \log n)$. Derive a recurrence for $T(n)$. (You do not have to solve the recurrence, because we have done that in class.)

**Hint:** The input only consists of $k$ and $x$. The matrix $H_k$, which has $n^2$ entries, is not given as part of the input. Since you are aiming for an $O(n \log n)$-time algorithm, you cannot compute all entries of the matrix $H_k$.

**Solution:** We will write the vector $x$ as

$$x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}.$$

Algorithm $\text{MULT}(k, x)$ is a recursive algorithm and does the following:

- If $k = 0$, return the vector $(x_1)$.

- Assume that $k \geq 1$. 

1
– Split the vector $x$ into two vectors $x'$ and $x''$, both of length $n/2 = 2^{k-1}$:

\[
x' = \begin{pmatrix} x_1 \\ \vdots \\ x_{n/2} \end{pmatrix}
\]

and

\[
x'' = \begin{pmatrix} x_{1+n/2} \\ \vdots \\ x_n \end{pmatrix}.
\]

– Run MULT($k - 1, x'$) and let the output be $y'$.

– Run MULT($k - 1, x''$) and let the output be $y''$.

– Compute the vector

\[
y = \begin{pmatrix} y' + y'' \\ y' - y'' \end{pmatrix}.
\]

– Return the vector $y$.

Let $T(n)$ denote the running time of algorithm MULT($k, x$), where $n = 2^k$. If $k \geq 1$, there are two recursive calls, both of which take time $T(n/2)$, whereas the rest of the algorithm takes $O(n)$ time. Thus, we obtain the “merge-sort recurrence”

\[
T(n) = \begin{cases} 
\text{constant} & \text{if } n = 1, \\
2 \cdot T(n/2) + O(n) & \text{if } n \geq 2.
\end{cases}
\]

We have seen in class that this recurrence solves to $T(n) = O(n \log n)$.