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

$$H_0 = (1)$$

and for $k \geq 1$,

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

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

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

(1.1) Let $k \geq 0$ be an integer and let $n = 2^k$. How many entries does the matrix $H_k$ have? Express your answer in terms of $n$.

Solution: We first determine the number of rows in the matrix $H_k$. Observe that $H_0$ has $1 = 2^0$ row. For $k \geq 1$, the number of rows in $H_k$ is twice the number of rows in $H_{k-1}$. By a straightforward induction, it follows that the number of rows in $H_k$ is equal to $2^k$.

By the same argument, the number of columns in the matrix $H_k$ is equal to $2^k$. Thus, the number of entries in $H_k$ is equal to

$$2^k \cdot 2^k = n \cdot n = n^2.$$

(1.2) Describe a recursive algorithm BUILD that has the following specification:

```
Algorithm BUILD(k):
Input: An integer $k \geq 0$.
Output: The matrix $H_k$.
```

For any positive integer $n$ that is a power of 2, say $n = 2^k$, let $T(n)$ be the running time of your algorithm BUILD($k$). Derive a recurrence for $T(n)$. Use the Master Theorem to give the solution to your recurrence.

Solution: We obtain the algorithm directly from the recurrence that is used to define the matrix $H_k$.
Algorithm BUILD($k$):
if $k = 0$
then return the matrix (1)
else $X = \text{BUILD}(k - 1)$;
$Y = -X$;
return the matrix \[
\begin{pmatrix}
X & X & Y \\
X & Y & X
\end{pmatrix}
\]
endif

Let $n \geq 2$; thus, $k \geq 1$. Algorithm BUILD($k$) generates one recursive call BUILD($k - 1$), which takes $T(n/2)$ time. The number of entries in $X$ is equal to $(n/2)^2 = O(n^2)$. Thus, the matrix $Y$ can be constructed in $O(n^2)$ time. Finally, in $O(n^2)$ time, three copies of $X$ and one copy of $Y$ can be combined to obtain the output of BUILD($k$). This shows that

$$T(n) = T(n/2) + O(n^2).$$

We are going to apply the Master Theorem: We have $a = 1$, $b = 2$, and $d = 2$. Since $d > \log_b a$, the Master Theorem tells us that $T(n) = O(n^2)$.

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

Describe a recursive algorithm MULT that has the following specification:

Algorithm MULT($k, x$):
Input: An integer $k \geq 0$ and a column vector $x$ of length $n = 2^k$.
Output: The column vector $H_k x$ (having length $n$).
Running time: must be $O(n \log n)$.

Explain why the running time of your algorithm is $O(n \log n)$. You are allowed to use the Master Theorem.

Hint: The input only consists of $k$ and $x$. The matrix $H_k$ is not given as part of the input.

Solution: An obvious algorithm first constructs the matrix $H_k$, by running algorithm BUILD($k$). Then it computes the product $H_k x$ using the definition of multiplication. Each of these steps takes $O(n^2)$ time. Since we are only allowed to spend $O(n \log n)$ time, we must compute $H_k x$ without constructing the entire matrix $H_k$. Of course, we can do this, because of the recursive definition of $H_k$.

We will write the column vector $x$ as

$$x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}.$$
• If \( k = 0 \), return the vector \((x_1)\).

• Assume that \( k \geq 1 \).
  
  – 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 \( \text{MULT}(k - 1, x') \) and let the output be \( y' \).
  
  – Run \( \text{MULT}(k - 1, x'') \) 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 \( \text{MULT}(k, x) \), 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”

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

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

Alternatively, we can use the Master Theorem to solve this recurrence: We have \( a = 2 \), \( b = 2 \), and \( d = 1 \). Since \( d = \log_b a \), the Master Theorem tells us that \( T(n) = O(n \log n) \).