Note: You are supposed to give proofs to the time bounds of your algorithms. Read the questions carefully before attempting to solve them.

1. Prove or disprove:
   - (8 points) If $f(n) = O(g(n))$ and $g(n) = O(h(n))$, then, $f(n) = O(h(n))$.
   - (8 points) $n^3 = O\left(n^{2\sqrt{\log n}}\right)$.
2. (17 points) Input is an array $A[1 : n]$ of arbitrary real numbers. The array could only be of one of the following two types: 1) **Type I:** $A$ has $\frac{n}{3}$ elements that are in the range $[1, 2)$, another $\frac{n}{3}$ elements in the range $[2, 3)$, and another $\frac{n}{3}$ elements in the range $[3, 4]$; or 2) **Type II:** $A$ has $\frac{n}{3}$ elements that are in the range $[2, 3)$, another $\frac{n}{3}$ elements in the range $[3, 4]$, and another $\frac{n}{3}$ elements in the range $(4, 5]$. Present a Las Vegas algorithm that determines the type of the array in $\tilde{O}(\log n)$ time.
3. (17 points) Assume that UCONN wants to keep records of its students and employees such that the following operations can be performed on the records:

- **Find_Name(HID)**: Return the name of the person whose HuskyOne ID is *HID*; and
- **Find_HID(Name)**: Return the HuskyOne ID of the person whose name is *Name*.

Present a data structure for keeping the records that will take $O(\log n)$ time to perform each of the above operations, $n$ being the total number of students and employees at UCONN. You can use $O(n)$ space.
4. (17 points) Present a data structure for real numbers that can perform the following operations:

1) Insert($x$): Insert the element $x$ into the data structure;
2) FindMin(): Return the minimum element that has been inserted into the data structure thus far;
3) FindMax(): Return the maximum element that has been inserted into the data structure thus far; and
4) FindMean(): Return the average value of all the elements that have been inserted into the data structure thus far.

Each operation should take $O(1)$ time.
5. Solve the following recurrence relations:

(a) (8 points)

\[
T(n) = \begin{cases} 
1 & n \leq 3 \\
27T\left(\frac{n}{3}\right) + n^4 & n > 3 
\end{cases}
\]

(b) (8 points)

\[
T(n) = \begin{cases} 
1 & n \leq 2 \\
T(n - 1) + \log_e n & n > 2 
\end{cases}
\]
6. (17 points) Input is a sequence $X_1, X_2, \ldots, X_{\log n}$ where each $X_i$ is a set of $\frac{n}{\log n}$ arbitrary real numbers. These sets need not be in sorted order. It is given that there exists a unique pair of sets $X_i$ and $X_j$ that have a common element. (No other pair of sets share any element). The problem is to identify $X_i$ and $X_j$. Present an $O(n \log n)$ time algorithm to solve this problem.