1. Prove or disprove:

   (a) (8 points) \( n^n = O\left(2^{(\log n)^{1.5}}\right) \).

   (b) (8 points) If \( f(n) = \Theta(n^4) \) and \( g(n) = \Theta(n^4) \), then \( f(n) - g(n) = \Theta(n^4) \).
2. (18 points) Input are two sorted arrays $A[1 : n]$ and $B[1 : n]$. These arrays can only be one of two types:

**Type I:** $A \cap B = \emptyset$

**Type II:** $|A \cap B| = n^{2/3}$.

Present a Monte Carlo algorithm that takes two arrays $A$ and $B$ and determines the type of these arrays. Your algorithm should run in time $O(n^{1/3} \log^2 n)$. Show that the output of your algorithm will be correct with high probability. (Fact: $(1 - x)^{1/x} \leq 1/e$ for any $0 < x < 1$.)
3. (16 points) A department has to keep records of its employees such that the following operations can be performed on the records:

- **Find_Name(SSN)**: Return the name of the person whose social security number is **SSN**; and
- **Find_SSN(Name)**: Return the social security number 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 number of persons in the department. You can use $O(n)$ space.
4. (a) (8 points) Solve the recurrence relation:

\[ T(n) = \begin{cases} 
1 & \text{if } n < 4 \\
64T \left(\frac{n}{4}\right) + n^3 & \text{if } n \geq 4 
\end{cases} \]

(b) (8 points) Solve the recurrence relation:

\[ T(n) = \begin{cases} 
1 & \text{if } n < 8 \\
T(n^{1/3}) + \log n & \text{if } n \geq 8 
\end{cases} \]
5. (16 points) An array has the following elements: \( k_1, k_2, \ldots, k_\ell, q_1, q_2, \ldots, q_m \), where the sequence \( k_1, k_2, \ldots, k_\ell \) is monotonically increasing and the sequence \( k_\ell, q_1, q_2, \ldots, q_m \) is monotonically decreasing. Also, \( \ell + m = n \) and \( \ell \) is unknown. Present an \( O(\log n) \) time algorithm to determine \( \ell \). For example, if the input is 2, 5, 11, 24, 8, 1, the answer is 4.
6. (18 points) Input is a sequence \( R, S_1, S_2, \ldots, S_{n_1}, T_1, T_2, \ldots, T_{n_2} \) of sequences of arbitrary real numbers. Here \( n_1 = 2\sqrt{\log n} \) and \( n_2 = \frac{n}{\log n} \). Each \( S_i \) is sorted and has \( \frac{n}{3\times2\sqrt{\log n}} \) elements, for \( 1 \leq i \leq n_1 \). Each \( T_i \) is also sorted and has \( \log n \) elements, for \( 1 \leq i \leq n_2 \). \( R \) may not be in sorted order and the number of elements it has is \( \frac{2}{3}n - \frac{n}{\log n} \). \( R \) has many duplications such that the number of distinct elements in it is \( 2\sqrt{\log n} \).

Present an \( O(n\sqrt{\log n}) \)-time algorithm to produce a sorted sequence of all the elements in all of these \((1 + n_1 + n_2)\) sequences.