1. Use the adversary lower bound technique to prove a lower bound on the time complexity of the problem of searching if \( x \) is in a sorted array \( A \) (of \( n \) elements) when using a comparison-based algorithm in which the algorithm can compare \( x \) with \( A[i] \) for any \( i \in \{0, 1, \ldots, n-1\} \), and will be told either that \( x < A[i] \), \( x = A[i] \), or \( x > A[i] \). The output from this algorithm should either be an index \( i \) into \( A \) for which \( A[i] = x \), or an indication that \( x \) is not in the array.

2. Use the adversary lower bound technique to prove a lower bound on the time complexity of a comparison-based algorithm for the following problem: You are given a sorted array \( A \) (of \( n \) elements) and two elements \( x \) and \( y \) where \( x \leq y \). The algorithm is required to compute how many elements in \( A \) are less than both \( x \) and \( y \), how many elements of \( A \) are between \( x \) and \( y \) (inclusive), and how many elements of \( A \) are bigger than both \( x \) and \( y \). Note that \( x \) and \( y \) are not necessarily in \( A \).

3. Suppose you are given the task to sort one thousand 32-bit keys. You have decided to use radix sort for this problem and want to decide how many bits each radix sort digit. Which is best among having 1 bit per radix sort digit, 4 bits per radix sort digit, 8 bits per radix sort digit or 16 bits per radix sort digit? You are provided with a counting sort procedure with exact time complexity of \( 5n + 4k \). Show your work.

4. Give the asymptotically fastest algorithm you can to sort \( n \) integers in the range of 0 to \( (n^4) - 1 \). You should give a very clear and complete high-level description of your algorithm. Be sure to analyze the time complexity of your algorithm as a function of \( n \). You are NOT restricted to use a comparison sorting algorithm (although are welcome to if you want).