1. Prove the following function

\[ T(n) = 10n^3 - 100n^2 + 1000n \]

is \( \Theta(n^3) \).

(a) Prove \( O(n^3) \):

(b) Prove \( \Omega(n^3) \):
2. Fibonacci sequence is recursively defined as: $F_1 = 1$, $F_2 = 1$, and $F_n = F_{n-1} + F_{n-2}$, $n \geq 3$.

(a) Compute and tabulate $F_n$ for $n = 1$ to 12.

(b) Prove by induction the following upper bound for $F_n$.

$$F_n \leq 2^{n-1}, \quad n \geq 1.$$  

(c) Prove by induction the following lower bound for $F_n$.

$$F_n \geq 2^{n/2-1}, \quad n \geq 1.$$
3. Consider the following divide-and-conquer algorithm (pseudocode). The initial call for an array \( A[0..n-1] \) is \( \text{SECRET}(A, 0, n-1) \).

```plaintext
Boolean \text{SECRET}([], int left, int right) {
    1. if \((left == right)\) return(\text{FALSE}); // Case of \( n = 1 \).
    2. if \((A[left] \neq A[right])\) return(\text{TRUE});
    3. int \( m = \lfloor (left + right)/2 \rfloor \);
    4. return (\text{SECRET}(A, left, m) \lor \text{SECRET}(A, m + 1, right)) // This is OR.
}
```

(a) Explain precisely what this program accomplishes. Prove your claim by induction.

(b) Let \( f(n) \) be the worst-case number of key comparisons for an array of size \( n \). (A key-comparison is the operation in line 2, which is a comparison between two array elements. The operation in line 1 is not a key comparison.) Write a recurrence for \( f(n) \), assuming that \( n \) is a power of 2. Find the exact solution by repeated substitution.
4. Given a sorted array $A[0..n-1]$ of 0’s and 1’s. The array consists of some number $k$ of zeros followed by $n-k$ ones, for some integer $k$, $0 \leq k \leq n$.

(a) Describe an efficient algorithm which takes the array $A$ as input and finds $k$. Your algorithm must have time complexity asymptotically better than $n$. Analyze the time complexity of your algorithm.

(b) Write the pseudocode for your algorithm.
5. The following recursive algorithm uses a **divide-and-conquer** technique to find the maximum element in an array of size $n$. The initial call for an array $A[0..n-1]$ is $\text{Findmax}(A, 0, n)$.

```c

dtype Findmax(dtype A[], int i, int n)
{
    // $i$ is the starting index, and $n$ is the number of elements.
    dtype Max1, Max2;
    if ($n == 1$) return $A[i]$;
    $Max1 = \text{Findmax}(A, i, \lfloor n/2 \rfloor)$; //Find max of the first half
    $Max2 = \text{Findmax}(A, i + \lfloor n/2 \rfloor, \lceil n/2 \rceil)$; //Find max of the second half
    if ($Max1 \geq Max2$) return $Max1$;
    else return $Max2$;
}
```

Let $f(n)$ be the worst-case number of *key comparisons* for finding the max of $n$ elements.

(a) Assuming $n$ is a power of 2, write a recurrence relation for $f(n)$.

(b) Find the solution of the recurrence by repeated substitution.

(c) Now apply the alternative method of proof by induction to show that the solution of the recurrence is $f(n) = An + B$ and find the constants $A$ and $B$. 
