1. Prove using induction that \( n \) is \( O(2^n) \).

2. Prove that \( 2^{n^2} \) is not \( O(5^n) \). Do not use any theorems about Big-Oh that you might happen to know other than the definitions.

3. Solve the following recurrence. Give a tight bound, i.e., express your answer using the \( \Theta \) notation. Assume that \( T(n) = 1 \), when \( n = 1 \).

\[
T(n) = T(n - 1) + \frac{1}{n}
\]

4. Consider the following code fragment

```java
for(int i=1;i<=n;i=2*i)
    for (int k = i; k >0; k = k/2)
        print('*');
```

a. Compute the number of stars printed as a function of \( n \). You can assume that \( n = 2^m \).

b. Give a \( \Theta \)-bound.

5. Modify the quicksort algorithm so that its running time is \( O(n \log n) \) in the worst case. You may assume that all elements are distinct.

6. Suppose that you have a “black box” worst-case linear-time median subroutine. Give a simple, linear-time algorithm that solves the selection problem for any arbitrary order statistic, i.e., given an unsorted array \( A \) containing \( n \) elements and an integer \( i \), in \( O(n) \) time, your algorithm should return the element in \( A \), which is the \( i^{th} \) smallest element in \( A \). Your algorithm must use the black-box median-finding subroutine. You may assume that \( i \) lies within the bounds of the input array \( A \) and that \( n \) is a power of 2. Justify your answer.
Some Useful Facts

1. \( \lg n = \log_2 n \)
   \( \ln n = \log_e n \)

2. Below are some formulas that may come handy.
   
   - \( \sum_{i=1}^{n} i = \frac{n(n+1)}{2} \)
   - \( \sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6} \)
   - \( \sum_{i=1}^{n} i^3 = \frac{n^2(n+1)^2}{4} \)
   - \( \sum_{i=0}^{n} c^i = \frac{c^{n+1}-1}{c-1}, c \neq 1 \)
   - \( \sum_{i=0}^{\infty} c^i = \frac{1}{1-c}, |c| < 1 \)
   - \( \sum_{i=1}^{\infty} c^i = \frac{c}{1-c^2}, |c| < 1 \)
   - \( \sum_{i=0}^{\infty} ic^i = \frac{c}{(1-c)^2}, |c| < 1 \)
   - \( H_n = \sum_{i=1}^{n} \frac{1}{i} = \ln n + O(1) \)

3. **Simplified Master Theorem.** Let \( a \geq 1, b > 1 \) be constants and let \( T(n) \) be the recurrence
   
   \[ T(n) = aT\left(\frac{n}{b}\right) + \Theta(n^k) \]
   
   defined for \( n \geq 0 \) (we assume that \( n \) is a power of \( b \), though this does not make a difference in asymptotic analysis). The base case, \( T(1) \) can be any constant value.

   Then
   
   **Case 1:** if \( a > b^k \), then \( T(n) \in \Theta(n^{\log_b a}) \).
   
   **Case 2:** if \( a = b^k \), then \( T(n) \in \Theta(n^{k \log_b n}) \).
   
   **Case 3:** if \( a < b^k \), then \( T(n) \in \Theta(n^k) \).