1. Order the following functions by growth rate (in non-decreasing order).

\[ n, n^{1.3}, n^2, n \log n, n \log \log n, n(\log n)^2, n \log(n^2), 2/n, 2^n, 2^{n/2}, 100, n^3, 1.2^n \]

2. Show that to find the second largest one of a list of \( n \) numbers, we need at least \( n - 2 + \lceil \log_2 n \rceil \) comparisons.

   **Hint:** We cannot determine the second largest element without having determined the largest element. Thus, the analysis can be done by the following:
   (1) Show that at least \( n-1 \) comparisons are necessary to find the largest element.
   (2) Show that there is always some sequence of comparisons which forces the second largest one to be found in \( \lceil \log_2 n \rceil \) additional comparisons.

3. Analyze the following procedure:

```plaintext
1 for i <- 0 to n
2 do for j <- i to n
3 do for k <- j to n
4 do print "A"
```

Prove that the running time of this procedure is \( O(n^3) \)
4. One sorting algorithm BUBBLE-SORT is introduced during class. Its running time is $O(n^2)$. Here is another sorting algorithm:

**INSERTION-SORT(A)**

1. for $j \leftarrow 2$ to length[A]
2. do key $\leftarrow A[j]$
3. i $\leftarrow j - 1$
4. while $i > 0$ and $A[i] > key$
5. do $A[i+1] \leftarrow A[i]$
6. i $\leftarrow i - 1$
7. $A[i+1] \leftarrow key$

(a) Please analyze it (about its running time).
(b) Try to design another simple sorting algorithm and give an analysis like (a). (The algorithm is described in pseudocode.)