1. (a) Find the exact solution of the following recurrence by using the repeated substitution method. Assume \( n \) is a power of 2.

\[
T(n) = \begin{cases} 
2T(n/2) + n^2, & n \geq 2 \\
0, & n = 1 
\end{cases}
\]

(b) Prove by induction that the solution of following recurrence is:

\[
T(n) = \begin{cases} 
4T(n-1) + 2^n, & n \geq 1 \\
0, & n = 0 
\end{cases}
\]

is: \( T(n) = A4^n + B2^n \) and find the constants \( A \) and \( B \).
2. A binary tree is called proper if every non-leaf node has exactly two children. Let $n$ be the total number of nodes in a proper binary tree. (Note that $n$ must be an odd integer.) And let $h$ be the height of the tree. (For $n = 1$, we define the height to be 0.)

(a) What is the minimum possible value of the height $h$ in terms of $n$? Explain.

(b) What is the maximum possible value of the height $h$ in terms of $n$? Prove your answer by induction.

(c) For $n = 11$, give two examples of proper binary trees, one with the minimum height and one with the maximum height.
3. This problem is to analyze the worst-case number of key comparisons for a simple 2-level sorting algorithm described below. Let \( n = p^2 \) be the number of elements to be sorted.

2.0 Divide the elements into \( p \) sets, each of size \( p \).
2.1 Sort the elements within each set using insertion sort.
2.2 Let \( M[i] \) be the largest element in the sorted set \( i \). Find the maximum value of all \( M[i], 1 \leq i \leq p \). Output this MAX element, while deleting it from its set.
2.3 Repeat step 2.2 in a similar fashion for finding the second MAX, third MAX, etc.

(a) What is the worst-case number of key comparisons to sort each set of \( p \) elements in step 2.1? Explain.

(b) Give a reasonable upper bound for the worst-case number of key comparisons to find the MAX in each iteration of step 2.2.
   Hint: You can find a reasonable upper bound by observing that each subsequent iteration takes no more than the first iteration.

(c) Use (a) and (b) to find a reasonable upper bound for the worst-case number of key comparisons (in terms of \( n \)) for the entire algorithm.
4. This problem is to design and analyze an efficient divide-and-conquer algorithm for finding the minimum and maximum elements in an array of size $n$. Assume $n$ is a power of 2.

(a) Write a recursive procedure (function) for such an algorithm. Provide comments.

(b) Let $C(n)$ be the worst-case number of key comparisons for this procedure, where $n$ is the number of elements. Write a recurrence for $C(n)$ and find the exact solution.
5. (a) Starting with an empty AVL tree, perform the following sequence of INSERT operations. (Be sure to clearly show any rebalancing needed.)

20, 40, 25, 30, 50, 60, 45, 70

(b) Show the final AVL tree below. Then, delete elements 20 and 30.