Instruction: There are five problems in this exam. Some of them consist of several subproblems. Make sure you look through them carefully. Please write legibly and justify all your answers.

1. **Growth Rates.** Order the following functions by growth rates:
   \[ n^2, 4^n, 2^{\log n}, 2^{2^n}, n \log n, \sqrt{n}. \]
   Please show your solution.

2. **Short questions.**
   a. Algorithms A and B are sorting algorithms that run in \( O(n \log n) \) and \( O(n^2) \)
      time respectively. When implemented on randomly generated data sets of size
      \( n < 100 \), it turns out that algorithm B ran faster. Why is this scenario possible?
   b. Explain why the maximum item in a binary heap must be at one of the leaves.
   c. Describe the properties of a \( B^+ \)-tree? When should this data structure be used?

3. **Linked Lists.**
   a. Given two sorted linked lists \( L_1 \) and \( L_2 \), describe a procedure that outputs
      \( L_1 \cup L_2 \). Your algorithm should run in \( O(|L_1| + |L_2|) \) time.
   b. Let us call your algorithm in (a) as \( \text{UNION}(L_1, L_2) \). How would you use
      \( \text{UNION} \) to compute for the union of \( n \) sorted linked lists \( L_1, L_2, \ldots, L_n \)?
      Describe the running time of your algorithm in terms of \( n \) and the size of the largest list,
      \( l_{\text{max}} \). That is, \( |L_i| \leq l_{\text{max}} \) for \( i = 1, \ldots n \).

4. **Binary Search Trees.**
   a. Suppose \( n \) keys are stored in a binary search tree \( T \). How would you sort the \( n \)
      keys in \( O(n) \) time?
   b. Suppose that we have numbers between 1 and 1000 in a binary search tree and want
      to search for the number 363. The following sequences of numbers cannot be
      the sequence of nodes examined. WHY?
      i. 925, 202, 911, 240, 912, 245, 363.
      ii. 935, 278, 347, 621, 299, 392, 359, 363.
      Can you generalize your reasoning to arbitrary sequences of numbers?
   c. Let \( T \) be a binary search tree whose keys are distinct. Let \( x \) be a leaf node,
      and \( y \) be its parent. Show that \( key[y] \) is either the smallest key in \( T \) larger than
      \( key[x] \) or the largest key in \( T \) smaller than \( key[x] \).
5. Let $S = \{x_1, x_2, \ldots, x_n\}$ be a set of distinct real numbers. Let $y$ be a real number as well. Our problem is to find two elements $x_i$ and $x_j$ in $S$ so that $x_i + x_j = y$, if such a pair exists.

a. Describe an $O(n^2)$ algorithm that will solve the problem in a brute force way.

b. This time, describe a different algorithm for the problem that runs in $O(n \log n)$ time. (Hint: Store the items of $S$ in a data structure so you can search through the items more efficiently.)