1. The following is the pseudo-code for a sorting procedure known as Bubble sort for sorting an array of \( n \) integers in ascending order.

   \hspace{2cm} \textbf{for} i = n-1 \textbf{downto} 1 \textbf{do}
   \hspace{2.5cm} \{ \text{ swapped := false;}
   \hspace{3cm} \textbf{for} j = 1 \textbf{to} i \textbf{do}
   \hspace{4cm} \textbf{if} A[j] > A[j+1] \textbf{then} swap them and set swapped to true;
   \hspace{4cm} \textbf{if} swapped \text{ is false} \textbf{then} halt;
   \}

   (a) Run bubble sort on the sequence 10 12 8 9 5 7.
   (b) Prove that Bubble sort correctly sorts its input.
   (c) What is the worst-case input for bubble sort? Use it to derive a lower bound on the time complexity of bubble sort in the worst case.
   (d) What is the best-case input for bubble sort? What is the time complexity of bubble sort for sorting this best-case input?

2. Run the Mergesort Algorithm described in class on the sequence 8 6 4 5 3 7.
   If \( n \) is odd let the left set have size \( \lfloor n/2 \rfloor \) and the right side have size \( \lceil n/2 \rceil \).

3. You are given an (implicit) infinite array \( A[1, 2, 3, \ldots] \).
   You are told that, for some unknown \( n \), the first \( n \) items in the array are positive integers sorted in increasing order while, for \( i > n \), \( A[i] = \infty \).
   Give an \( O(\log n) \) algorithm for finding the largest non-\( \infty \) value in \( A \).

4. \( a_1, a_2, \ldots, a_n \) is a sequence that has the following property:
   \textit{There exists some} \( k \) such that
   \[ \forall i : 1 \leq i < k, \quad a_i > a_{i+1} \quad \forall i : k \leq i < n, \quad a_i < a_{i+1}. \]
   Such a sequence is \textit{unimodal} with unique minimum \( a_k \).
   Design an \( O(\log n) \) algorithm for finding \( k \).
5. Extra Problem. *The limits of comparison-based lower bounds*  
Let $S = \{x_1, x_2, \ldots, x_n\}$ be a set of integers or real numbers. Let $y_1, y_2, \ldots, y_n$ be the same numbers sorted in increasing order. The MAX-GAP of the original set is the value
\[
\text{Max}_{1 \leq i < n}(y_{i+1} - y_i).
\]

Using a more advanced form of the $\Theta(n \log n)$ proof of the lower bound for sorting it can be proven that calculating MAX-GAP requires $\Theta(n \log n)$ operations if only comparisons and algebraic calculations are used. In this problem, we will see that, if the floor (truncate) operator $\lfloor x \rfloor$ can also be used, the problem can be solved using only $O(n)$ operations!

- Find $y_1$ and $y_n$, the minimum and maximum values in $S$.
- Let $\Delta = \frac{y_n - y_1}{n - 1}$. Let $B_i$ be the half-closed half-open interval defined by
  \[
  B_i = [y_1 + (i - 1)\Delta, y_1 + i\Delta)
  \]
  for $i = 1, 2, \ldots, n - 1$ and set $B_n = \{y_n\}$.
- Prune $S$ as follows. For every $B_i$ throw away all items in $S \cap B_i$ except for the smallest and largest. Let $S'$ be the remaining set.
- Find the Max-Gap of $S'$ by sorting $S'$ and running through the items in $S'$ in sorted order. Output this value.

Prove that this algorithm outputs the correct answer and show that every step can be implemented in $O(n)$ time.