1. Die in der Vorlesung gezeigte Version von Quicksort funktioniert nicht sehr gut, wenn viele der zu sortierenden Schlüssel gleich sind. In dieser Frage geht es um dieses Phänomen und mögliche Verbesserungen.

   (a) (6 Punkte) Zeigen Sie, wenn alle n Schlüssel in einem zu sortierenden Feld gleich sind, dann benötigt die in der Vorlesung angegebene Version von Quicksort $\Theta(n^2)$ Zeit.

   (b) (14 Punkte) Ändern Sie die Partitionsroutine (in Spezifikation und/oder Implementierung), sodass für die in (a) angegeben Eingabe eine wesentlich bessere Laufzeit erreicht wird. Welche Laufzeit können Sie erzielen?

   (c) (20 **Extra**punkte) Zeigen Sie, dass mit einer geeigneten Version von Quicksort ein Feld mit n Schlüsseln, die aber nur d verschiedene Werte annehmen, in erwarteter Zeit $O(n \log d)$ sortiert werden kann.

2. (15 Punkte)
   Für eine Folge $a = \langle a_1, \cdots, a_n \rangle$ von Schlüsseln sei $F(a)$ die Anzahl der "Fehlstellen" in a, also die Anzahl der Paare a_i, a_j mit $1 \leq i < j \leq n$ und $a_i > a_j$. Wenn die Folge a schon aufsteigend sortiert ist, dann gilt $F(a) = 0$. Wenn sie absteigend sortiert ist und alle Schlüssel verschieden sind, dann gilt $F(a) = \binom{n}{2}$.

   Zeigen Sie, dass die Laufzeit von InsertionSort, wenn angewandt auf eine Folge a der Länge n, in $O(n + F(a))$ liegt.

   Warum ist das n in dieser Laufzeitschranke notwendig?

3. (15 Punkte)
   Wenn Mergesort auf ein Feld angewandt wird, gibt es die Schwierigkeit, dass sich zwei sortierte Teilfelder nicht so ohne Weiteres verschmelzen ("mergen") lassen, ohne zusätzlichen Speicher zu verwenden und Teilfelder zu kopieren. In der Mergesortversion aus der Vorlesung wird z.B. jeder Schlüssel zweimal irgendwohin zugewiesen, zuerst beim Kopieren, dann beim Verschmelzen.

   Geben Sie eine oder mehrer Möglichkeiten an, wie dieser Kopieraufwand (und vielleicht auch der Bedarf an extra Speicher) deutlich reduziert werden kann.

   Wir erwarten hier neben einer allgemeinen Erklärung, wie Ihr Ansatz funktioniert, auch explizite Programme in Pseudocode mit Kommentaren und Analysen.

4. (10 Punkte)

   (a) Geben Sie ein Paar von Funktionen f und g an, sodass weder $f \in O(g)$ noch $g \in O(f)$ gilt. Beweisen Sie die Richtigkeit Ihrer Antwort.

   (b) Gibt es so ein Paar von Funktionen auch, wenn zusätzlich die Bedingung besteht, dass sowohl f als auch g streng monoton steigend ist?