1. (10 Punkte) Betrachten Sie InsertionSort, QuickSort, MergeSort und Heapsort. Geben Sie für jede diese Sortiermethoden an, ob sie stabil ist, oder nicht. Beweisen Sie ihre Antworten.

2. (20 Punkte)
   Die Prozedur Stable-Partition(A, i, j, x) soll das Teilfeld $A[i..j]$ so umstellen, dass $A[i..p]$ nur Schlüssel enthält die kleiner gleich x sind, und $A[p+1..j]$ nur Schlüssel größer gleich x enthält. Dabei sollen allerdings keine zwei gleiche Schlüssel im Teilfeld in ihrer Reihenfolge vertauscht werden. Die Zahl p soll natürlich als Ausgabe auch zurückgegeben werden.

   (a) (4 Punkte) Welche Laufzeit können Sie erreichen, wenn beliebig viel zusätzlicher Speicher vorhanden ist?

   (b) (16 Punkte) Welche Laufzeit können Sie erreichen, wenn Sie nur konstant viel zusätzlichen Speicher verwenden? (Je besser die Laufzeit, desto mehr Punkte)

3. (20 Punkte)
   Es sei $B = \{0, 1, \ldots, b-1\}$ für irgendeine natürliche Zahl $b > 1$. Für ein k-Tupel $x = (x_k, x_{k-1}, \ldots, x_1) \in B^k$ sei $'x$ das $(k-1)$-Tupel $(x_{k-1}, \ldots, x_1)$ und x' das $(k-1)$-Tupel $(x_k, x_{k-1}, \ldots, x_2)$. Wir betrachten die übliche lexikographische Ordnung auf Tupeln, also die signifikanteste Komponente an der linksten Stelle.

   Es sei X eine Menge von n Tupeln aus B^k, die sortiert werden soll. Das Radixsort-Verfahren beruht auf der Einsicht, dass wenn die $x \in X$ schon bezüglich $'x$ vorsortiert sind, dann reicht ein stabiles Sortieren auf der linkesten Komponente x_k um X insgesamt lexikographisch sortiert zu bekommen. Mit Hilfe von CountingSort lässt sich das in $O(n + b)$ Zeit bewerstelligen.

   (a) Nehmen wir nun an, die Menge X sei bezüglich x' vorsortiert. Entwerfen Sie einen Algorithmus, der X lexikographisch sortiert. Welche Laufzeit können Sie erreichen?

   (b) Nehmen wir an, die Menge X sei bezüglich x' in ein Feld $A[1..n]$ vorsortiert und dazu gäbe es ein Feld $E[1..n-1]$ von Bits mit $E[i] = 1$ falls $A[i]' = A[i+1]'$.
   Wie könen Sie diese Zusatzinformation ausnutzen, um X lexikographisch zu sortieren? Welche Laufzeit könen Sie erreichen?

   In dieser Aufgabe gehen wir davon aus, dass auf die i-te Komponente eines Tupels in konstanter Zeit zugegriffen werden kann und dass Elemente aus B in konstanter Zeit vergleichen werden könen.

4. (10 Punkte)
   Sie ziehen auf eine einsame Insel und dürfen 50 kg Bücher mitnehmen. Sie besitzen n Bücher und wissen von jedem Buch sein Gewicht. Sie möchten möglichst viele Bücher mitnehmen.

   Geben Sie einen Algorithmus an, der bestimmt, wie viele und welche Bücher Sie mitnehmen. Ihr Algorihtmus sollte nur $O(n)$ Laufzeit brauchen. Sie sollen dabei **nicht** annehmen, dass die Gewichte natürliche Zahlen sind, bzw. leicht als solche dargestellt werden können. Ihr Algorithmus sollte also nur folgende Operationen auf Gewichten verwenden: der Größe nach vergleiche, addieren, subtrahieren.

   (In modernen Zeiten bringt man natürlich keine 50 kg physiche Bücher auf die Insel, sondern 16 Gigabyte digitale Bücher, oder was auch immer. Das Problem bleibt das gleiche.)