**Aufgabe 4.1** (4 Punkte)
Sortieren Sie die Folge $\langle e_1, \ldots, e_{10} \rangle = \langle 3, 1, 4, 1, 5, 9, 2, 6, 5, 3 \rangle$ mit Hilfe von Mergesort. Stellen Sie die rekursiven Aufrufe, die Zwischenergebnisse und den Ablauf von *merge* im äußersten Aufruf ähnlich wie im Abb. 5.3 im Buch dar.

**Aufgabe 4.2** (4 Punkte)
Erstellen Sie in Pseudocode eine arraybasierte und rekursiv arbeitende Implementation von Mergesort namens $MergeSort(A, B, l, r)$, der bei jedem Aufruf als Parameter ein Eingabearray $A$, ein Hilfsarray $B$ und zwei ganze Zahlen $l$ und $r$ übergeben werden. Zu Beginn enthält das Eingabearray eine Anzahl zu sortierender Einträge, die fortlaufend an den Positionen $1, 2, \ldots$ gespeichert sind. Beide Arrays sollten zur Speicherung aller Zwischenergebnisse und zum Aufbau der Ausgabe benutzt werden. Die Parameter $l$ und $r$ geben in jedem (rekursiven) Aufruf die Positionen des linken bzw. rechten Randes desjenigen Teilarrays von $A$ an, innerhalb dessen sortiert werden soll. Kommentieren Sie nötigenfalls Ihr Programm ausreichend!

**Aufgabe 4.3** (4 Punkte)
Gegeben seien vier Elemente $e_1, \ldots, e_4$. Geben Sie einen Vergleichsbaum (der nur die beiden Vergleiche $\leq$ und $>$ verwendet) an, der die vier Elemente nach höchstens fünf Vergleichen ermittelt. Beginnen Sie mit dem Vergleich $e_1 \leq e_2$. Begründen Sie, warum die Maximalzahl benötigter Vergleiche hier mindestens fünf betragen muss.