.

**Aufgabe 7.1**

Die Klasse `MyPriorityQueue` aus der Vorlesung soll um eine Methode

```
public void increasePriority(int index,int newPriority)
```

erweitert werden, die es ermöglicht, die Priorität des Eintrags, welcher im Array `heap` an Stelle `index` gespeichert ist, auf einen neuen Wert `newPriority` zu setzen. Die bisherige Priorität des Eintrages soll aber nur dann geändert werden, wenn `newPriority` echt größer ist als die bisherige Priorität. Überlegen Sie sich, wie Sie die gewünschte Funktionalität umsetzen würden. Gehen Sie insbesondere darauf ein, wie Sie sicherstellen, dass die Heapeigenschaft erhalten bleibt, bzw. wie Sie diese ggfs. wieder herstellen. Setzen Sie Ihre Idee dann in einer entsprechenden Methode um und speichern Sie die so erweiterte Klasse `MyPriorityQueue` in einer Datei `MyPriorityQueue.java`. (6 Punkte)

**Aufgabe 7.2**

Schreiben Sie eine Methode analog zu der Methode `mergeArrays` aus der Vorlesung, die die Elemente zweier sotierter Folgen ganzer Zahlen mischt, die in einer verketteten Liste gespeichert sind.

```
static MyLinkedList mergeLL(MyLinkedList e,MyLinkedList f),
```

Die resultierende sortierte Gesamtfolge soll ebenfalls als verkettete Liste zurückgegeben werden. Die neue Methode soll Teil eines Programms `MergeLL.java` sein, in dem Sie anhand von zwei Beispiel-Listen mit den Elementen $\{1, 5, 7\}$ und $\{0, 2, 3, 6, 8\}$ überprüfen, daß Ihre Methode korrekt arbeitet. Die Elemente der zusammengefügten Liste sollen ausgegeben werden. (8 Punkte)

**Zusatzaufgabe 7.3**

Erweitern Sie das in der Vorlesung besprochene Programm `MatchBrackets.java` um eine Methode:

```
static boolean checkExpression(String ausdruck)
```

Der übergebene String kann außer öffnenden und schließenden runden Klammern auch noch alle möglichen anderen Zeichen enthalten. Es soll gestestet werden, ob die enthaltenen runden Klammern in der Reihenfolge, in der diese von links nach rechts vorkommen, einen wohlgeformten Klammerausdruck bilden. (3+Punkte)