Aufgabe 1. Erstelle ein Programm `uniq`, dass zwei Dateinamen als Kommandozeilenargumente erhält, die erste Datei zeilenweise als Integer-Variable einliest und ohne doppelte Einträge in die zweite Datei schreibt (in irgendeiner Reihenfolge).

Aufgabe 2. Diese Aufgabe läuft auf die Implementierung des Merge-Sort Algorithmus hinaus.

a) Implementiere eine Funktion `merge`, die zwei bereits sortierte (eventuell verschieden große) Arrays als Argumente erhält, diese zu einem sortieren Array kombiniert und dieses zurück liefert.

b) Die Funktion `mergesort` selbst soll ein Array in zwei (möglichst gleich große) Teilarrays zerlegen, sich für diese Teilarrays selbst aufrufen und danach die dann sortierten Teilarrays mit der `merge`-Funktion kombinieren. Erhält die Funktion ein Array mit keinem oder einem Element so belässt es dieses Array wie es ist, dann ist es nämlich bereits sortiert.

Hier als Tipp ein Vorschlag für die Signaturen der beiden Funktionen:

```
1 int *merge(int *list1, int n, int *list2, int m);
2 void mergesort(int *list, int n);
```

Aufgabe 3. Gegeben sei eine Datei, in der ausschließlich Zahlen stehen. In der ersten Zeile stehe eine natürliche Zahl, die angibt, wie viele Zeilen noch folgen. Die noch folgenden Zeilen bestehen ebenfalls nur aus einer Zahl.

a) Schreibe ein Programm, dass diese Datei einliest, die Zahlen sortiert und die Datei mit der sortierten Liste überswchreibt.

b) Modifiziere dein Programm nun so, dass in der ersten Zeile nicht mehr stehen muss, wie viele Zeilen noch folgen.

Aufgabe 4. Erweitere dein Matrizen-Modul um eine Funktion, die Matrizen aus einer Datei lesen kann. Ein Format für diese Dateien darfst du dir selbst ausdenken.