



Aufgabe 1. Implementiere einige Funktionen um mit quadratischen Matrizen umzugehen:

- a) Eine Funktion, die Speicher für eine quadratische Matrix allokiert, eine um ihn freizugeben, eine um sie in eine Datei zu schreiben und eine, um sie zur Einheitsmatrix zu initialisieren :

```
1 double **matrix_alloc(int n);  
2 void matrix_free(double **A, int n);  
3 void matrix_fprint(FILE *fp, double **A, int n);  
4 double **matrix_id(double **A, int n);
```

- b) Eine Funktion um eine Matrix zu transponieren.
- c) Eine Funktion, die zwei solche Matrizen miteinander multipliziert und eine neue Matrix zurück gibt. Für zwei $n \times n$ -Matrizen $A = (a_{ij})$ und $B = (b_{ij})$ ist $A \cdot B = C = (c_{ij})$ durch $c_{ij} = \sum_{k=1}^n a_{ik}b_{kj}$ definiert.

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 sortierten 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.
- c) Besorge dir die Datei `daten.h`, sortiere das darin definierte Array und schreibe es sortiert in eine Datei.

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);
```