Aufgabe 1. Denke dir einen sinnvollen Namen aus für ein Modul, dass Vektorrechnung auf $\mathbb{R}^n$ implementiert. Vektoren sollen `double`-Arrays mit der Länge n sein. Implementiere die nachfolgenden Funktionen:

a) eine Funktion, die genügend Speicher für einen Vektor reserviert und einen Pointer darauf zurück gibt

b) Vektoraddition

c) Vektorsubtraktion

d) Produkt eines Vektors mit einer skalaren Größe

e) Skalarprodukt zweier Vektoren

f) eine Funktion, die einen Vektor auf der Konsole aus gibt

Zur Verdeutlichung hier ein Beispiel einer Funktion, die einen Vektor aufnimmt und ihn mit dem Nullvektor initialisiert:

```
1 void vector_make0(double *v, int n) {
2         int i;
3         for(i=0; i<n; i++) v[i] = 0;
4 }
```

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, diese Teilarrays rekursiv durch einen weiteren Aufruf von `mergesort` sortieren und die so sortierten Teilarrays mit der `merge`-Funktion kombinieren.

   Wenn die Funktion ein Array der Länge 1 oder 0 übergeben bekommt, so tut die Funktion nichts, da das Array bereits sortiert ist: Dies beendet die Rekursion.

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. 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 auszugeben und eine um sie zur Einheitsmatrix zu initialisieren (das ist die Matrix mit 1en auf der Hauptdiagonale und 0en sonst):

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

b) Eine Funktion um eine Matrix zu transponieren (d.h. an der Hauptdiagonale "zu spiegeln")

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 $A \cdot B = C = (c_{ij})$ über folgende Formel definiert:

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$